60483000

CONTROL DATA CORPORATION

FORTRAN EXTENDED VERSION 4 TO FORTRAN VERSION 5 CONVERSION AID PROGRAM VERSION 1 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1
SCOPE 2

REVISION RECORD

Revision	Description
A (03/23/79)	Original release at PSR level 485.
B (08/24/79)	Revised to include support of SCOPE 2 operating system.
C (03/04/83)	Revised to include technical corrections.

REVISION LETTERS I, O, Q, AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Front Cover	-
Title Page	-
ii	C
iii/iv	C
v/vi	B
vii	B
viii	B
1-1	A
2-1 thru 2-3	A
2-4	B
2-5	B
2-6	A
2-7	B
2-8	B
3-1	A
3-2	B
3-2.1	B
3-3 thru 3-6	A
3-7 thru 3-10	B
4-1	B
4-2	A
4-3	B
A-1	B
A-2	A
B-1 thru B-5	B
C-1	A
D-1 thru D-12	A
D-13	C
D-14	C
D-15 thru D-18	A
Index-1	B
Comment Sheet	C
Mailer	-
Back Cover	-

PREFACE

This manual describes the FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aid Program, which is designed to produce FORTRAN Version 5 source statements from FORTRAN Extended Version 4 input. Wherever possible, the output from the conversion program complies with the American National Standards Institute FORTRAN (called FORTRAN 77) as described in ANSI X3.9-1978. If the conversion program produces any nonstandard statements, the user has the option with the FORTRAN Version 5 compiler to flag those statements.

The reader should have knowledge of FORTRAN Extended Version 4 and should only convert programs that have compiled and run successfully under FORTRAN Extended Version 4.

The conversion program runs under the following operating systems:

NOS 1 for CONTROL DATA® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems

NOS/BE 1 for CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems

SCOPE 2 for CONTROL DATA®CYBER 170 Model 176, CYBER 70 Model 76, and 7600 computer systems

Related information can be found in the publications listed below.

Publication	Publication Number
FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN Version 5 Reference Manual	60481300
Modify Reference Manual	60450100
SCOPE Version 2 Reference Manual	60342600
Update Version 1 Reference Manual	60449900

CDC Manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

INTRODUCTION 1

The FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aid Program (Conversion Aid) is designed to produce translations that conform as closely as possible to the FORTRAN Version 5. FORTRAN Version 5 is based on the ANSI FORTRAN 77 language. Some of the major differences in FORTRAN 77 include IF-THEN-ELSE constructs, the PARAMETER statement, and CHARACTER data. To make programs more portable, the standard now incorporates such extensions as PRINT, ENTRY, and alternate return.

FORTRAN Version 5 (FORTRAN 5) includes the complete FORTRAN 77 standard, but has additional special features. The FORTRAN 5 compiler can, as an option, flag any nonstandard statements. Obsolete or little-used features have not been included in FORTRAN 5. A few features have changed functionally, but not syntactically. These include the computed GOTO and the DO loop. The C$ DEBUG package has been discontinued, but CYBER Interactive Debug is available and provides a powerful debug facility.

Over 50 differences exist between FORTRAN Extended Version 4 (FORTRAN 4) and FORTRAN Version 5, but most of these differences can be automatically converted with little or no manual effort. The conversion program flags any statements it cannot convert.

Input for the conversion program can be either a standard or sequenced source file or an Update/Modify COMPILE file. The Conversion Aid produces a listing and a source output file. The complete listing contains the original programs, the converted programs, the Update/Modify directives, and messages. The messages identify statements that must be manually changed, the type of translation performed, or the type of errors. The conversion program has a variety of options for processing.

The Conversion Aid converts only valid FORTRAN 4 statements as described in the FORTRAN Extended Version 4 Reference Manual, and does not convert undocumented features. Programs depending on a particular FORTRAN 4 implementation might not produce the same results after conversion.

The Conversion Aid does not perform extensive syntax analysis. Since its purpose is to detect language differences, it might ignore certain syntax errors. Input to the conversion program should consist only of programs that have compiled and executed correctly under FORTRAN 4.

Valid FORTRAN Extended Version 4 (FORTRAN 4) source statements can be input to the FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aid Program (Conversion Aid) in the following forms:

FORTRAN 4 source images on a specified file (standard or sequenced)

COMPILE file from Update/Modify

The output from the Conversion Aid consists of a listing and a source output file containing FORTRAN Version 5 (FORTRAN 5) statements. The listing provides information about the conversion, including messages, and the source output file contains the converted program or the changes necessary for input to Update/Modify. The F45 control statement is used to call the Conversion Aid, specifying the type of input to be used and the type of output to be produced.

INPUT TO CONVERSION AID

The Conversion Aid reads a single input file and processes it according to the parameters on the F45 control statement. The I parameter is used for the input file name. The SI parameter specifies whether the file is sequenced. The input file can have more than one program unit; the Conversion Aid recognizes the end of one program unit and the beginning of another.

The input file can be in one of three formats: sequenced, standard, or COMPILE. Sequenced lines have line numbers in columns 1 through 5. Standard lines do not have line numbers in the first five columns, but might have statement labels. The statements for standard lines are coded in columns 7 through 72; any information can be in columns 73 through 80. Lines in COMPILE format have the 90-column format; this format is the same as standard format, except that information also appears after column 80. A file cannot contain programs in both sequenced format and COMPILE format.

SEQUENCED INPUT

A file is treated as sequenced if the SI parameter is selected, or if columns 1 through 5 on the first line of input contain a number and SI=0 is not specified. See the time-sharing mode and SEQ parameter in the FORTRAN Extended Version 4 Reference Manual. Both standard format and COMPILE format are considered unsequenced. If the file is unsequenced and any column after 80 contains a nonblank character, the file is assumed to be a COMPILE file.

UPDATE/MODIFY INPUT

The Conversion Aid can read a COMPILE file and produce an output file that can be used to update the old program library from which the COMPILE file was generated.

When the PO=M parameter is selected and the input is a COMPILE file, the Conversion Aid generates an Update/Modify directive for each statement that needs to be translated. The directive is a command to delete the original statement and insert the translated statement.

UPDATE/MODIFY OUTPUT

For a manual change, the original statement is listed as a replacement line, but is not translated. The Conversion Aid generates a directive to delete the original statement and to replace it with the same untranslated statement. If the user changes the statement before a library update, then the user must remove the untranslated statement from the source output file and replace it with the new statement.

Corrective Directives

The Conversion Aid generates, in most cases, a directive for each change. Because it is impossible, however, to determine the deck names from the COMPILE file, DECK directives are not generated; MODIFY users must generate these directives.

Common Directives

A programmer can use three directives to update a source program library containing common decks. The directives have the characters C$ in columns 1 and 2, and a keyword beginning in column 7. The three keywords are BEGCOM, ENDCOM, and LIST. Without these directives, every time a common deck appears in a COMPILE file, the Conversion Aid translates the common deck and provides directives to update the old program library. If a common deck appears more than once, the Conversion Aid generates duplicate directives. The duplicate directives will produce errors during an update of the old program library.

Using the LIST directive, the programmer can direct the Conversion Aid to translate the common decks only once. Then the old program library will be updated correctly.

BEGCOM and ENDCOM mark the beginning and ending of a common deck in the program library. The LIST directive is used in the calling program. If LIST is omitted, or if LIST(C=0) is specified, the lines between BEGCOM and ENDCOM are not converted. If LIST(C) or LIST(C=1) appears, the lines are converted. To convert any common deck, LIST must precede *CALL, the directive on the program library to call a common deck. Examples of common deck conversion are shown in figures 2-1 and 2-2.

```
*COMDECK NAME1
C$      BEGCOM
          .
          .
          .
(source lines)
          .
          .
          .
C$      ENDCOM

*COMDECK NAME2
C$      BEGCOM
          .
          .
          .
(source lines)
          .
          .
          .
C$      ENDCOM
```

Figure 2-1. Common Decks in a Program Library Example

```
        PROGRAM P
C$      LIST(C)
          .
          .
          .
*CALL   NAME1           (converts NAME1)
          .
          .
          .
        END

        PROGRAM Q
          .
          .
          .
*CALL   NAME1           (does not convert NAME1)
          .
          .
          .
C$      LIST(C=0)
*CALL   NAME1           (does not convert NAME1)
C$      LIST(C)
          .
          .
          .
*CALL   NAME2           (converts NAME2)
          .
          .
          .
        END
```

Figure 2-2. Programs with Common Decks Example

OUTPUT FROM CONVERSION AID

The Conversion Aid produces two output files: a listing and a source output file. The listing is formatted for printing; a line has a maximum of 137 characters, including a carriage control character in column 1. The source output file is formatted into 80-column lines. The output lines retain the original format as much as possible. Statements that are correct for FORTRAN 5 remain unchanged. When a statement must be changed, leading and embedded blanks are retained. If an item in a statement is deleted, the blanks following are also deleted. If a statement is added in standard format, it begins in column 7 and ends in column 72 (in sequenced format, the statement ends in column 80). If a statement must be continued, it is broken at a delimiter and continued on the next line, beginning in column 7.

The conversion program can produce either sequenced or unsequenced output from sequenced or unsequenced input. The SO parameter controls sequenced output. When the Conversion Aid converts sequenced input to unsequenced output, the output statements are in standard format. When the Conversion Aid converts unsequenced input to sequenced output, the output statements are in sequenced format.

Wherever possible, embedded comments are retained in the FORTRAN 4 statements. If the original FORTRAN 4 statements are totally restructured so that the comments have questionable value, the comments are repositioned in front of the original statements. Comments between statements remain in their original positions.

The Conversion Aid generates a comment header for each program unit. The header appears in both the listing and the source output file. The header information states which version of the Conversion Aid converted the program units and the version of FORTRAN 5 for which the converted programs are valid.

LISTING

The user can select one of four output formats for the listing, or can suppress the listing altogether. One of the choices, the full listing, includes a copy of the input file. The other formats include only statements changed or added by the Conversion Aid and statements requiring manual action.

In the listing, a source statement is followed by any Conversion Aid messages applying to it. The messages are of three types: automatic conversion, manual action, and error diagnostics. The messages are described in appendix B.

For each action that the Conversion Aid takes, it prints a symbol at the end of the input line to indicate what the action was. The meanings for the three symbols are:

- -I- Statement inserted.
- -R- Statement replaced previous statement.
- -D- Statement deleted.

The Conversion Aid writes a message summary at the end of each program unit.

Full Listing

The full listing has two parts. The first part is a copy of the input file. The second part is in one of two formats, depending on the input. If the input is a COMPILE file, the second part is exactly the same as the Update/Modify listing. If the input file contains sequenced or standard lines, the second part contains:

Unchanged lines

Changed and added lines

Lines requiring manual action

Error diagnostics

A statistical summary of Conversion Aid processing

Short Listing

The short listing is like the second part of the complete listing, except that the unchanged statements are not included. The output contains:

Changed and added lines

Lines requiring manual action

Error diagnostics

A statistical summary of Conversion Aid processing

Update/Modify directives are not listed, even if the input is a COMPILE file.

Error Listing

The error listing contains:

Statement lines requiring manual action

Error diagnostics

Update/Modify directives are not listed, even if the input is a COMPILE file.

Modification Listing

The modification listing is exactly the same as the second part of the full listing, except that Update/Modify directives precede the statements. This format of the listing is used only when the input is a COMPILE file. The modification listing contains:

Changed and added lines

Lines requiring manual action

Error diagnostics

A statistical summary of Conversion Aid processing

Note that if the user ignores lines requiring manual action, the lines are deleted and reinserted, so that there is no net change. If the user wishes to change the line, the replacement line can be used as a template to create a new line.

SOURCE OUTPUT FILE

The user can select one of three formats for the source output file, or can suppress creation of the file. The source output file does not include any Conversion Aid messages except for the comment header.

Full Source Output File

The full source output file is generated when input is a sequenced or standard file. This output is a complete program unit, ready for compilation under FORTRAN 5 except for any statements requiring manual inspection or action. The full file contains:

Comment header generated by the Conversion Aid

All unchanged lines

Changed lines

Added lines

Lines requiring manual action or inspection

If the input file is a COMPILE file and the user requests a full source output file, the Conversion Aid generates a modification file.

Short Source Output File

The short source output file consists only of changed statements rather than complete program units. The short source output file is used for manually updating FORTRAN 4 source programs and contains:

Changed lines

Added lines

Lines requiring manual action or inspection

Modification Source Output File

The modification source output file is generated when the input is a COMPILE file. If the input file is sequenced or in standard format and a modification source output file is requested, the Conversion Aid generates a message but no output. The output is intended for an Update/Modify program library and contains directives.

Because it is impossible to determine the DECK names from COMPILE file output, the MODIFY user must insert the necessary DECK directives.

COMPASS SUBROUTINES

COMPASS subroutines can appear in the output without change. If LO=F is selected, COMPASS subroutines are written directly to the listing. If PO=F is selected, COMPASS subroutines are written to the source output file.

F45 CONTROL STATEMENT

The Conversion Aid is called with a control statement that conforms to operating system syntax; the statement cannot be continued on another line. The control statement takes one of the following three forms:

F45.

F45,p_1,p_2,...,p_n.

F45(p_1,p_2,...,p_n)

The optional parameters, p_i, can be in any order.

A variety of options can be specified in the parameter list. No parameter can be used more than once. For any omitted parameters, default values are supplied. Specific default values are given with the description of each parameter. An invalid or duplicate parameter causes the job to terminate abnormally. Comments can follow the terminating right parenthesis or period.

More than one program can be converted by a single F45 call. When several programs are to be converted, the programs should follow each other without separation. The Conversion Aid recognizes the end of each program unit.

In the following parameter descriptions, lfn denotes a file name, which is one through seven letters and digits beginning with a letter.

CC - COMMENT CONTROL

FORTRAN 4 recognizes a statement with a $ in column 1 as a comment; FORTRAN 5 does not. The Conversion Aid changes the $ to a C or an *, as specified by the CC parameter. The options are:

omitted

Change $ indicating a comment line to C.

CC

Same as CC=*.

CC=C

Same as omitted.

CC=*

Change $ indicating a comment line to *.

CI - CORRECTION IDENTIFIER FOR UPDATE/MODIFY

The CI parameter specifies an update correction identifier for a run with a COMPILE file as input. If LO=M or LO=F is also specified, the identifier is written on the listing. If PO=M or PO=F is specified, the identifier is written on the source output file. Unless at least one of the options LO=M, LO=F, PO=M, or PO=F is specified, the CI parameter is ignored. The options are:

omitted

Generate an Update/Modify directive of the form:

*IDENT dddhhmm

The ddd is the number of the day of the year. The hh is the hour of the day and mm is the minutes.

CI

Same as omitted.

CI=0

Do not generate an *IDENT directive, even if LO=M, LO=F, PO=M, or PO=F is specified.

CI=idname

Generate an Update/Modify directive of the form:

*IDENT idname

The idname is the correction identifier to be assigned to this set. Because this parameter appears on a control statement, the operating system limits idname to seven characters (any combination of letters and digits).

DD - DELETE C$ DIRECTIVES

Both FORTRAN 4 and FORTRAN 5 provide compiler directives that begin with a C$ in columns 1 and 2. FORTRAN 5 does not recognize the FORTRAN 4 directives; the FORTRAN 4 directives would produce an error under FORTRAN 5. The Conversion Aid converts the FORTRAN 4 directives as specified by the DD parameter. The options are:

omitted

Delete all statements with a C$ in columns 1 and 2.

DD

Same as omitted.

DD=0

Convert statements with a C$ in columns 1 and 2 to comments by replacing the $ with a blank.

ET - EXIT TERMINATION

The ET parameter specifies exit termination activity. The options are:

omitted

Terminate normally, no matter what conditions exist in the input file.

ET

Terminate normally only if none of the following conditions exist: FORTRAN syntax errors, statements requiring manual action (not just inspection), or requests for Update/Modify output files when input is not a COMPILE file. The Conversion Aid aborts if any of these conditions exist, and system error exit processing then takes over.

ET=0

Same as omitted.

I - SOURCE INPUT FILE

The I parameter specifies the source input file. The options are:

omitted

Same as I=INPUT.

I

Same as I=COMPILE.

I=lfn

Read source programs from the file named lfn.

L - LISTING FILE

The L parameter specifies the listing file. The options are:

omitted

Same as L=OUTPUT.

L

Write listing on the file named LIST.

L=0

Do not produce listing.

L=lfn

Write listing on the file named lfn.

LO - LISTING OPTIONS

The LO parameter specifies listing options. The options are:

omitted

Produce a short listing containing:

Translated and added lines

Lines requiring manual action

Conversion Aid messages and error diagnostics

Note that the Conversion Aid does not list Update/Modify directives, even if input is a COMPILE file.

LO

Produce a full listing with two parts: a copy of the input file and a listing that depends on the type of input. The first part of the listing is always produced and consists of a copy of the input file. If the input file is standard or sequenced, the second part of the listing contains:

Lines that require no change

Changed and added lines with the related Conversion Aid message

Lines requiring manual action with the related Conversion Aid message

Messages and error diagnostics

If input is a COMPILE file, the second part of the listing contains:

Generated Update/Modify directives

Translated and added lines with the related Conversion Aid message

Lines requiring manual action with the related Conversion Aid messages

Messages and error diagnostics

LO=S

Same as omitted.

LO=F

Same as LO.

LO=E

Produce an error listing containing:

Lines requiring manual action and related Conversion Aid messages

Error diagnostics

Note that the Conversion Aid does not list Update/Modify directives, even if input is a COMPILE file.

LO=M

Produce a modification listing containing:

Generated Update/Modify directives

Translated and added lines

Lines requiring manual action

Messages and error diagnostics.

LO=M is meaningful only if the input is a COMPILE file.

MC - MASTER CONTROL CHARACTER

The special characters +, -, /, and = must be specified with the delimiter $; the special characters are master control characters used in Update/Modify. For example, to specify /, then MC=$/$; to specify $, then use two consecutive $ such as MC=$$$$.

omitted	The master control character is *.
MC	Same as omitted.
MC=$char$	The master control character is char.

MD - MACHINE-DEPENDENT USAGES

The MD parameter specifies machine-dependent usages. The options are:

omitted

Ignore the machine-dependent constructs.

MD

Issue a manual change message for each statement that contains at least one machine-dependent construct. The Conversion Aid recognizes the following machine-dependent constructs:

Hollerith data

Shifts and masks

Intrinsic functions dealing with bit manipulation: XOR, OR, AND, COMPL, and LOCF

Octal and hexadecimal data

ENCODE and DECODE statements

BUFFER IN and BUFFER OUT statements

If the MD parameter is specified, the Conversion Aid flags statements containing these constructs, warning the programmer that they might have to be changed if the program is ever transferred to another computer.

P - SOURCE OUTPUT FILE

The P parameter specifies the source output file. The options are:

omitted

Do not produce any source output.

P

Write source output on the file named PUNCH.

P=0

Same as omitted.

P=lfn

Write source output on the file named lfn.

PD - PRINT DENSITY

The PD parameter specifies print density. The options are:

omitted

Produce a listing with a print density of six lines per inch.

PD

Produce a listing with a print density of eight lines per inch.

PD=6

Same as omitted.

PD=8

Same as PD.

PO - SOURCE OUTPUT OPTIONS

This parameter selects the data to be included on the source output file. If the P parameter is not specified or if P=0 is specified, the Conversion Aid ignores this parameter and does not produce a source output file.

omitted

Produce a short source output file containing:

Lines requiring manual action

Changed lines

Added lines

The short source output file is intended for manually updating the original source decks. This output is not suited for direct input to the compiler.

PO

Produce a modification file. If the input is standard or sequenced, the Conversion Aid issues an error message and produces no output. If the input is a COMPILE file, the Conversion Aid produces a modification file containing:

Update/Modify directives to modify the old program library

Changed lines

Added lines

Lines that require manual action

Except for lines requiring manual action, modification output is suitable for modifying an old program library.

PO=S

Same as omitted.

PO=F

Produce a full source output file. If the input is a COMPILE file, PO=F is equivalent to PO=M. If the input is standard or sequenced, the Conversion Aid produces a complete file containing:

Input lines that need no translation

Changed lines

Added lines

Lines that require manual action

Except for lines requiring manual action, this output is suitable for direct input to the compiler.

PO=M

Same as PO.

SI - SEQUENCED INPUT

If the Conversion Aid performs automatic selection from the first statement of the input file, the selection is permanent for the entire run; SI is not reset for individual program units.

Wrong selection is made if the input is unsequenced and the first statement is labeled with a number in columns 1 through 5. Although legitimate, this case is unusual and can be accommodated by explicitly including SI=0 in the F45 control statement. The explicit value overrides the automatic selection.

omitted

Determine the input format from columns 1 through 5 of the first input line. If these columns contain a number, sequenced input is assumed. If columns 1 through 5 do not contain a number, standard input is assumed.

SI

Treat the input as sequenced. If columns 1 through 5 of any statements do not contain a sequence number, a diagnostic message is issued, and the statement is not translated. The user's selection of SI continues to be honored.

SI=0

Treat the input as standard. If columns 1 through 5 of any statement contain a sequence number, the number is assumed to be a statement label, and the user's selection of SI=0 continues to be honored.

SO - SEQUENCED OUTPUT

This parameter selects sequenced output; it is not meaningful for a COMPILE file. The sequence numbers for sequenced output are created according to the options n1, n2, and n3. The n1 option is the first sequence number, n2 is the increment, and n3 is the number of digits in the first output sequence number.

omitted

Determine the mode of the output file from the mode of the input file. If the input file is sequenced, omitting the SO parameter is equivalent to specifying SO, and the output files are sequenced. If the input file is unsequenced, omitting the SO parameter is equivalent to specifying SO=0, and the output files are unsequenced.

SO

Create sequenced output files, with the sequence numbers determined by the input file. If the input is unsequenced or if the input is sequenced but the first line of the input does not contain a sequence number, specifying SO is equivalent to specifying SO=10/10/5.

If the input is sequenced and the first line of the input contains a sequence number, n3 is set to the number of digits in the sequence number in the first line of the input file. If n3 is less than or equal to 2, n1 and n2 are set to 1. If n3 is greater than 2, n1 and n2 are set to 10.

SO=0

Create unsequenced output files.

SO=n1/n2/n3

All three values must be specified. The greatest number of digits that field n1 can have is five. The largest value that n1 can be is 99999. If 99999 is exceeded in an F45 run, the next sequence number output will begin from zero.

SUMMARY OF F45 PARAMETERS

A summary of the parameters for the F45 control statement is shown in table 2-1.

TABLE 2-1. F45 PARAMETER SUMMARY

Parameter	Use	First Default (keyword omitted)	Second Default (keyword only)	Notes
CC	Comment Control	CC=C	CC=*	For comments beginning with $.
CI	Correction Identifier	CI=dddhhmm	CI=dddhhmm	Requires LO=M, LO=F, PO=M, or PO=F.
DD	Delete C$ Directives	DD	DD	Can be DD=0.
ET	Exit Termination	ET=0	ET	
I	Source Input File	I=INPUT	I=COMPILE	Can be I=lfn.
L	Listing File	L=OUTPUT	L=LIST	Can be L=lfn or L=0.
LO	Listing Options	LO=S	LO=F	Can be LO=M or LO=E.
MC	Master Control Character	MC=$*$	MC=$*$	Can be MC=$char$.
MD	Machine-Dependent Usages	Off	On	Issues manual change messages for the constructs.
P	Source Output File	P=0	P=PUNCH	Can be P=lfn.
PD	Print Density	PD=6	PD=8	Applies to listing.
PO	Source Output Options	PO=S	PO=M	Can be PO=F.
SI	Sequenced Input	Determine from first line	Sequenced Input	Can be SI=0 for standard input.
SO	Sequenced Output	Depends on SI parameter	Sequenced Output	Can be SO=0 for standard output, or can be SO=n1/n2/n3.

The FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aid Program (Conversion Aid) can convert most FORTRAN Extended Version 4 (FORTRAN 4) statements to FORTRAN Version 5 (FORTRAN 5) statements. Whenever the Conversion Aid performs an automatic conversion, a message describes what conversion has taken place.

LANGUAGE ELEMENTS

Some changes have been made for FORTRAN 5 nonexecutable statements. The features affected are comments, continuation lines, multiple statements, blank lines, octal constants, Hollerith constants, and arrays. Although most of these conversions are trivial, the long Hollerith constant conversion might not produce the same results under FORTRAN 5 as in the original FORTRAN 4 program.

COMMENT LINES

FORTRAN 5 does not recognize a comment line beginning with a $ in column 1. Under FORTRAN 4, any of the characters C, *, or $ in column 1 indicates a comment line. Only the characters C and * are valid for FORTRAN 5. If a $ in column 1 occurs, the Conversion Aid changes it to either a C or an *, depending on the control statement parameter CC.

CONTINUATION LINES

A FORTRAN 4 continuation line can contain any characters in columns 2 through 5, and any character other than C, *, or $ in column 1. FORTRAN 5 requires that a continuation line have blanks in columns 1 through 5. The Conversion Aid inserts blanks into columns 1 through 5 of a continuation line.

The Conversion Aid inserts blanks into columns 1 through 5 of a continuation line if the input file is not sequenced. For a sequenced file, the program inserts a sequence number in the first five columns.

MULTIPLE STATEMENTS

Multiple statements on a line are not allowed in FORTRAN 5; each statement must begin on a separate line. The Conversion Aid converts a line containing multiple statements separated by the $ delimiter into several lines, each containing a single statement. For example, the line:

```
A=B $ C=C+1. $ D=A+C
```

becomes the three lines:

```
A=B
C=C+1.
D=A+C
```

The converted statements all begin in the same column in which the original line began. The superfluous $ characters are discarded.

BLANK LINES IN A CONTINUATION

FORTRAN 4 interprets a blank source line as an initial statement line, which breaks a possible continuation sequence. FORTRAN 5 interprets a blank line as a comment line, which does not break a possible continuation sequence. This difference is significant when a blank line immediately precedes a continuation line. Figure 3-1 shows several ways blank lines are converted.

The lines:

```
A=B
(blank line)
+C=D
```

are converted to:

```
A=B
(blank line)
C=D
```

The lines:

```
A=B
(blank line)
+
+C=D
```

are converted to:

```
A=B
(blank line)
(blank line)
C=D
```

The lines:

```
A=B
+
+=D
```

remain unchanged.

Figure 3-1. Blank Lines in a Continuation

When the Conversion Aid encounters a blank source line, it continues reading source lines until it finds a nonblank line. If the nonblank line is a continuation line, the Conversion Aid converts it to an initial line. If the nonblank line becomes a blank line when it is converted to an initial line, the conversion program repeats the process until it finds either a nonblank initial line or a continuation line that is converted to a nonblank initial line.

OCTAL CONSTANTS

For FORTRAN 4, an octal constant has the form nnnnB. For FORTRAN 5, an octal constant has the form O"nnnn". In both cases, nnnn is an octal number with no more than 20 digits. The Conversion Aid converts nnnnB to O"nnnn". For example:

```
M=23B+5634B
```

becomes:

```
M=O"23"+O"5634"
```

LONG HOLLERITH CONSTANTS

A long Hollerith constant is a character string that is longer than 10 characters. FORTRAN 4 permits a long Hollerith constant bounded only by the length of a statement. FORTRAN 5 does not allow a Hollerith constant to exceed 10 characters. FORTRAN 5 provides a new data type, CHARACTER, to handle longer strings. A Hollerith constant under FORTRAN 5 can appear in three contexts: in an expression, in a DATA statement, or as an actual argument. The Conversion Aid treats these three cases differently.

In an expression, FORTRAN 4 ignores all characters after the leftmost 10 characters. The Conversion Aid truncates any long Hollerith constant in an expression to 10 characters. The resulting constant produces the same result under FORTRAN 5 as the original constant did under FORTRAN 4.

In a DATA statement, FORTRAN 4 allows a long Hollerith constant to initialize successive elements of an array. (See the FORTRAN Extended Version 4 Reference Manual for details.)

If a long Hollerith constant is the last item in a data list, the Conversion Aid breaks the long constant into a series of 10-character constants, possibly followed by one shorter constant. If the long constant is not the last constant in the data list, the Conversion Aid truncates the long constant to 10 characters. The user should check that the code generated by the Conversion Aid produces the same results as the original code, especially if a procedure depends on trailing zeros.

When used as an actual argument, a long Hollerith constant is not shortened. For the Hollerith forms nH or "..." (delimited constant), the Conversion Aid converts the constant to a type character constant with the form '...'. For a long constant with the form nR or nL, the Conversion Aid replaces the constant with an integer array name that it creates. The array name is:

```
Znnnnn
```

where nnnnn is a unique combination of digits and alphabetic characters.

In figure 3-2, several Hollerith constants are translated; messages are generated for any truncated character strings or for any portability problems.

ZERO-FILLED HOLLERITH CONSTANTS

Zero-filled Hollerith constants of the following form are no longer permitted in FORTRAN 5:

```
nRf
nLf
```

where:

n = unsigned decimal integer representing the number of characters in the string including blanks.

f = string of characters; must contain at least one character.

L = left-justified with binary zero fill.

R = right-justified with binary zero fill.

These forms are translated as follows:

nRf	nLf
becomes	becomes
R"f"	L"f"

For example:

3RABC	3LABC
becomes	becomes
R"ABC"	L"ABC"

UNSUBSCRIPTED ARRAY NAMES

FORTRAN 4 permits an unsubscripted array name to appear in any context where an array or an array element could appear. If the unsubscripted array name appears in a context where the name could not be interpreted as a reference to the entire array, FORTRAN 4 assumes that the missing subscripts are all one. If the array name appears with fewer subscripts than in the DIMENSION or type statement which defined the size of the array, FORTRAN 4 assumes that all the missing subscripts have a value of one.

```
INTEGER CHARS(2), STRING(3)
DATA CHARS/17R17CHARACTERSTRING/
STRING(1)=14HLONGCHARSTRING
STRING(2)="CHAR" "STRING"
STRING(3)="TENCHARAC" " " " " " "
CALL MESSAGE(21LTHIS PROGRAM RAN O.K.)
CALL MESSAGE(6HSTRING)
CALL MESSAGE(4LCHAR)
```

is converted to:

```
INTEGER CHARS(2), STRING(3)
INTEGER ZZZ001(3)
DATA CHARS/10R17CHARACTE,7RRSTRING/
DATA ZZZ001/10LTHIS PROGR, 10LAM RAN O.K, 1L/
STRING(1)= 10HLONGCHARST  (diagnostic message)
STRING(2)= "CHAR" "STRIN"  (diagnostic message)
STRING(3)= "TENCHARAC" " " (diagnostic message)
CALL MESSAGE(ZZZ001)
CALL MESSAGE("STRING")
CALL MESSAGE(4LCHAR)
```

Figure 3-2. Long Hollerith Conversion

FORTRAN 5 permits an unsubscripted array name to reference an entire array, but requires that any reference to a single array element have the same number of subscripts as in the DIMENSION or type statement defining the size of the array. This means that under FORTRAN 5, any unsubscripted array name refers to the entire array.

The Conversion Aid adds the implied subscripts to any array name appearing in a context not referring to the entire array. For example:

```
DIMENSION A(10), AA(2,4), AAA(5,5,5)
B= A
BB= AA(3)
BBB= AAA(3,2)
```

is translated to:

```
DIMENSION A(10), AA(2,4), AAA(5,5,5)
B= A(1)
BB= AA(3,1)
BBB= AAA(3,2,1)
```

EXPRESSIONS AND ASSIGNMENT STATEMENTS

The major differences in expressions and assignment for FORTRAN 5 include the order in which exponentiation factors are evaluated, and a restriction on complex operands in relational expressions. Abbreviations for logical expressions and signed typeless operands are not allowed. The Conversion Aid makes the necessary changes.

EXPONENTIATION

FORTRAN 4 interprets successive exponentiation factors from left to right; FORTRAN 5 interprets from right to left. In FORTRAN 4, A**B**C is evaluated as (A**B)**C. In FORTRAN 5, A**B**C is evaluated as A**(B**C). The Conversion Aid adds additional parentheses to make the FORTRAN 4 implied interpretation explicit. For example:

```
A=B**C**D
```

is converted to:

```
A=(B**C)**D
```

COMPLEX OPERANDS IN RELATIONAL EXPRESSIONS

In a relational expression, FORTRAN 4 permits either operand to be complex for any relational operator. If the operator is neither .EQ. nor .NE., FORTRAN 4 compares only the real parts of the operands. FORTRAN 5 permits an operand to be complex only if the operator is .EQ. or .NE. A relational expression with complex operands and any other operator produces an error.

When a relational expression includes one or more complex operands and the relational operator is neither .EQ. nor .NE., the Conversion Aid changes each complex operand to a real operand by using the intrinsic function REAL. For example, the statements:

```
COMPLEX C, CC
    .
    .
    .
IF(C.GT.CC) GO TO 15
```

would be converted to:

```
COMPLEX C, CC
    .
    .
    .
IF(REAL(C).GT.REAL(CC)) GO TO 15
```

LOGICAL EXPRESSIONS

FORTRAN 4 allows the abbreviations .A., .O., .N., and .X. for the logical operators .AND., .OR., .NOT., and .XOR.; it allows .T. and .F. for the truth values .TRUE. and .FALSE. FORTRAN 5 does not allow these abbreviations. The conversion program expands these abbreviations.

SIGNED TYPELESS CONSTANTS AND OPERANDS

FORTRAN 4 allows a typeless operand to be signed; FORTRAN 5 does not. A typeless operand is an octal constant, a Hollerith constant, or a result of one of the intrinsic functions SHIFT and MASK. In a DATA statement, FORTRAN 5 interprets a sign preceding a typeless constant as an operator and diagnoses an error. If the sign of the typeless operand in a DATA statement is plus, the Conversion Aid deletes the sign; if the sign is minus, the Conversion Aid replaces the operand with its 60-bit ones complement value. For example:

```
DATA A, B/11B, +222B/

DATA C/-333B/
```

is translated to:

```
DATA A, B/O"11", O"222"/

DATA C/O"7777777777777777444"/
```

Notice that the B suffix for octal constants is also changed.

In an assignment statement or statement function definition, the Conversion Aid interprets a unary plus or minus operator followed by a typeless operand as an unsigned constant prefixed by an operator, and converts the constant as indicated by the context. The statements:

```
WORD1= +222B
WORD2= -3HXYZ
WORD3= -333B
WORD4= -SHIFT(STRING,24)
```

are translated to:

```
WORD1= O"222"
WORD2= (.NOT. 3HXYZ)
WORD3= (.NOT. O"333")
WORD4= (.NOT. SHIFT(STRING,24))
```

For a minus sign, the conversion program replaces the sign with .NOT. and encloses the negated operand with parentheses. If the sign is a plus, the conversion program deletes the sign.

SPECIFICATION STATEMENTS

FORTRAN 5 does not accept the keyword TYPE or the abbreviated form DOUBLE for DOUBLE PRECISION in a declaration. Labeled common blocks cannot be identified by numbers. The Conversion Aid assigns unique name labels to the numbered common blocks. LEVEL statements cannot contain variable names unless the names refer to dummy arguments. The variable name is replaced with the corresponding common block name. The DATA statement syntax has been changed to reduce confusion between a list of two real constants and a complex variable. Redundant parentheses are prohibited.

TYPE STATEMENTS

FORTRAN 5 does not permit the keyword TYPE to precede the type declaration keyword in a type statement. The Conversion Aid deletes TYPE from any type statement where it appears.

FORTRAN 5 does not allow the abbreviation DOUBLE for DOUBLE PRECISION in a type statement. The Conversion Aid converts the declaration DOUBLE to the declaration DOUBLE PRECISION in any type statement. For example, a statement with both TYPE and DOUBLE:

```
TYPE DOUBLE D,DA
```

becomes:

```
DOUBLE PRECISION D,DA
```

NUMBERED COMMON BLOCKS

FORTRAN 4 allows numbered common blocks. FORTRAN 5 allows only blank or named common blocks. The Conversion Aid converts the number identifying a common block to a name derived from that number. First the program converts the decimal number to a base 36 number, then adds the letter Z as a prefix. The digit 0, letters A through Z, and digits 1 through 9 are arbitrarily used as base 36 digits. The generated common block names are unique with respect to each other but might duplicate variable names in the program.

For example, the lines:

```
COMMON/0/ A(4)
COMMON/100/ B(5)
COMMON/19/ C(100)
```

are converted to the lines:

```
COMMON/Z0/ A(4)
COMMON/ZB1/ B(5)
COMMON/ZS/ C(100)
```

LEVEL STATEMENT

The LEVEL statement specifies the storage level of dummy arguments and variables contained in common blocks. The LEVEL statement in FORTRAN 5 specifies the storage level of dummy arguments and common blocks, but not variables. The Conversion Aid changes a reference to a variable to a reference to the common block containing that variable. Dummy arguments are not changed. The Conversion Aid generates only one LEVEL statement reference to a common block, even if the FORTRAN 4 program contains LEVEL statements referencing several items from the same common block.

The Conversion Aid generates new LEVEL statements with common block names for the individually listed variables and arrays. A sample LEVEL statement conversion is illustrated in figure 3-3.

DATA STATEMENT

The FORTRAN 4 DATA statement differs from FORTRAN 5 in both the way the list can be identified and in the possible format of the list itself.

FORTRAN 4 and FORTRAN 5 both accept a DATA statement of the form:

```
DATA vlist/dlist/
```

FORTRAN 5 does not accept the alternate form of the DATA statement:

```
DATA (vlist=dlist)
```

The Conversion Aid converts this format to the standard format. In figure 3-4, several data statements are converted.

```
      PROGRAM MAIN
      COMMON /BLK2/ A, B, C
      COMMON /BLK3/ R(6)
      EQUIVALENCE (R,S)
      LEVEL 2, A, B, C
      LEVEL 3, S
          .
          .
          .
      SUBROUTINE SUB(D)
      COMMON /BLK2/ AA, BB, CC
      LEVEL 2, AA, D
      LEVEL 2, BB, CC
          .
          .
          .
```

is converted to:

```
      PROGRAM MAIN
      COMMON /BLK2/ A, B, C
      COMMON /BLK3/ R(6)
      EQUIVALENCE (R,S)
      LEVEL 2, /BLK2/
      LEVEL 3, /BLK3/
          .
          .
          .
      SUBROUTINE SUB (D)
      COMMON /BLK2/ AA, BB, CC
      LEVEL 2, /BLK2/, D
          .
          .
          .
```

Figure 3-3. Conversion of LEVEL Statements

```
      DIMENSION A(10), AA(7), AAA(3,3,3)
      INTEGER B
      COMPLEX C, CC(4)

      DATA (R=0.1), (RR=3.45)
      DATA (A(2)=2.2), (AA=7*(0.))
      DATA (((AAA(I,J,K),I=1,3), J=1,3), K=1,3)
     +=13*(0.3),14*(0.5))
      DATA (B=7H( ) ( ) ( ) ( )
      DATA (C, CC=(1.1,2.2), 4*((3.3,4.4)))
```

is converted to:

```
      DIMENSION A(10), AA(7), AAA(3,3,3)
      INTEGER B
      COMPLEX C, CC(4)

      DATA R/0.1/, RR/3.45/
      DATA A(2)/2.2/, AA/7*0./
      DATA (((AAA(I,J,K), I=1,3), J=1,3), K=1,3)
     +/13*0.3,14*0.5/
      DATA B/7H( ) ( ) ( ) (/
      DATA C, CC/(1.1,2.2), 4*(3.3,4.4)/
```

Figure 3-4. DATA Statement Conversion

The data list for the FORTRAN 4 DATA statement consists of one or more of the following forms separated by commas:

```
constant
(constant list)
rf*constant
rf*(constant list)
rf(constant list)
```

where rf is a repeat factor. Redundant parentheses are allowed. Under FORTRAN 5, the data list in the DATA statement consists of one or more of the following forms, separated by commas:

```
constant
rf*constant
rf(constant list)
```

Redundant parentheses are prohibited.

The FORTRAN 4 data list formats do not distinguish between a single complex constant and a constant list consisting of two real constants surrounded by parentheses.

FORTRAN 4 can interpret the form (constant list), where constant list is a pair of real constants, as either a single complex constant or two real constants. If the corresponding data item is a complex variable, the form (constant list) specifies a complex constant. If the corresponding data items are two real variables, the form (constant list) specifies two real constants. Under FORTRAN 5, the form (constant list) specifies a single complex constant; therefore, if the corresponding data items are two real variables, the Conversion Aid removes the redundant parentheses. The form:

```
(real constant, real constant)
```

is changed to:

```
real constant, real constant
```

The form ((constant list)) is a complex constant in FORTRAN 4. The Conversion Aid removes the redundant parentheses to produce a FORTRAN 5 complex constant with the form (constant list).

The FORTRAN 4 form rf*(constant list) always specifies a repeated list of two real constants; the form rf*((constant list)) specifies a repeated complex constant. For FORTRAN 5, the form rf*(constant list) specifies a repeated complex constant, and the form rf(constant list) specifies a repeated list of two real constants. The Conversion Aid converts the forms:

```
rf*(real constant, real constant)
rf*((real constant, real constant))
rf*(constant1, constant2, constant3)
```

to the forms:

```
rf(real constant, real constant)
rf*(real constant, real constant)
rf(constant1, constant2, constant3)
```

The Conversion Aid removes unnecessary parentheses. For example:

```
(constant1, constant2, constant3)
```

becomes:

```
constant1, constant2, constant3
```

FLOW CONTROL STATEMENTS

Flow control statements affect the execution of a program. The FORTRAN 4 statements that are changed or absent in FORTRAN 5 include computed GOTO, two-branch arithmetic IF and logical IF, PAUSE, STOP, and END.

For the computed GOTO, if the index variable has an integer value less than one or greater than the number of statement labels in parentheses, execution of the statement in FORTRAN 5 does not produce a fatal error. FORTRAN 5 does not support any two-branch IF statements. PAUSE, STOP, and END have minor changes.

COMPUTED GOTO STATEMENTS

If a computed GOTO statement has a variable or expression with an integer value less than one or greater than the number of statement labels in parentheses, FORTRAN 4 programs abort with a fatal error; programs compiled by FORTRAN 5, however, continue with the next executable statement following the computed GOTO.

To duplicate FORTRAN 4 behavior, the Conversion Aid adds a call to an error routine, GOTOER. The routine issues a message and aborts the program when the integer variable is outside the legal range. If the computed GOTO is not part of a logical IF statement, the Conversion Aid inserts the new statement CALL GOTOER immediately after the computed GOTO statement. For example:

```
GO TO (10, 11, 15, 20) JUMP
```

is converted to the two statements:

```
GO TO (10, 11, 15, 20) JUMP
CALL GOTOER
```

If the computed GOTO is part of a logical IF statement, the Conversion Aid inserts the new statement CALL GOTOER as part of the IF loop. For example:

```
IF (A.LT.B) GOTO (10, 11, 20) JUMP
```

is converted to the statements:

```
IF(A.LT.B) THEN
GOTO(10, 11, 20) JUMP
CALL GOTOER
ENDIF
```

The block IF construction is a FORTRAN 5 feature.

ARITHMETIC IF STATEMENTS

FORTRAN 4 allows the expression in an arithmetic IF statement to be complex; only the real part of the expression is tested. FORTRAN 5 does not allow the expression to be complex.

The Conversion Aid changes a complex expression in an arithmetic IF statement to a real expression by using the intrinsic function REAL. For example:

```
COMPLEX C, CC
      .
      .
      .
IF(C+CC)10,20,30
```

becomes:

```
COMPLEX C, CC
    .
    .
    .
IF(REAL(C+CC))10,20,30
```

TWO-BRANCH ARITHMETIC IF STATEMENTS

FORTRAN 5 does not permit a two-branch arithmetic IF statement. The Conversion Aid converts a two-branch arithmetic IF statement to a three-branch statement. For example:

```
IF(X-Y)15,17
```

becomes:

```
IF(X-Y)15,17,15
```

The first branch is used again as the third branch.

TWO-BRANCH LOGICAL IF STATEMENTS

FORTRAN 5 does not permit a two-branch logical IF statement. The Conversion Aid changes each two-branch logical IF statement to a standard form logical IF statement that produces the same result. For example:

```
IF(TEST)601,604
```

becomes:

```
IF(TEST)GOTO 601
GOTO 604
```

PAUSE AND STOP STATEMENTS

Both FORTRAN 4 and FORTRAN 5 allow the programmer to append a comment to a PAUSE or STOP. The comment is sent to the dayfile when the statement is executed. Under FORTRAN 4, the comment is specified by a Hollerith constant; under FORTRAN 5, the comment is specified by a character constant. The Conversion Aid changes each constant in a PAUSE or STOP statement to a FORTRAN 5 character constant. For example:

```
STOP "THE END"
```

becomes:

```
STOP 'THE END'
```

END STATEMENT

FORTRAN 4 diagnoses the absence of an END statement from the final program unit of a compiler input file and adds the END statement. FORTRAN 5 considers the absence of an END statement as a fatal error. The Conversion Aid inserts an END statement after the final program unit of the input file if none was included.

FORTRAN 5 does not allow an END statement to be continued. The Conversion Aid replaces a continued END statement with one that is complete in one line.

OUTPUT STATEMENTS

Certain differences exist between FORTRAN 4 and FORTRAN 5 output statements. Under FORTRAN 5, a WRITE statement requires a unit number; PRINT can only be used when data is sent to the OUTPUT file. Any Hollerith constant in a list-directed output statement must be changed to a character constant.

WRITE/PRINT/PUNCH STATEMENTS

FORTRAN 4 and FORTRAN 5 output statements PRINT, PUNCH, and WRITE all have the same effect when a unit number is specified. PRINT and WRITE have the same effect when no unit number is specified. FORTRAN 5 requires that WRITE be used for output whenever a unit number is specified, and that PRINT be used whenever no unit number is specified and data is to be sent to the file OUTPUT. Table 3-1 shows the conversion for output statements.

TABLE 3-1. CONVERSION OF OUTPUT STATEMENTS

FORTRAN 4 Statement	Converted to FORTRAN 5 Statement
WRITE fn	PRINT fn
WRITE fn,iolist	PRINT fn,iolist
WRITE *,iolist	PRINT *,iolist
PRINT (u,fn)	WRITE (u,fn)
PRINT (u,fn)iolist	WRITE (u,fn)iolist
PRINT (u,*)iolist	WRITE (u,*)iolist
PUNCH (u,fn)	WRITE (u,fn)
PUNCH (u,fn)iolist	WRITE (u,fn)iolist
PUNCH (u,*)iolist	WRITE (u,*)iolist

HOLLERITH CONSTANTS IN LIST-DIRECTED OUTPUT

Unlike FORTRAN 4, FORTRAN 5 does not accept Hollerith constants in the output list of a list-directed output statement. In FORTRAN 4, the Hollerith constant is displayed in integer form. The Conversion Aid converts such Hollerith constants to character constants so that each constant is printed in character form. For example:

```
PRINT *, 4HTEST
```

is converted to:

```
PRINT *, 'TEST'
```

and the line:

```
PRINT *, "TEST"
```

is converted to:

```
PRINT *, 'TEST'
```

INPUT/OUTPUT LISTS AND FORMAT STATEMENTS

The changes for FORTRAN 5 include eliminating redundant parentheses and requiring commas with edit descriptors; FORTRAN 5 also requires minor changes in edit descriptors.

REDUNDANT PARENTHESES IN INPUT/OUTPUT LISTS

FORTRAN 4 allows items in an input or output list to be surrounded with redundant pairs of parentheses. The redundant parentheses can result in ambiguous statements. Consider the statement:

```
PRINT *, (2.31,8.)
```

According to the syntax, the iolist could consist of either a single complex constant or two real constants. FORTRAN 4 assumes that a list of exactly two real constants surrounded by at least one pair of parentheses indicates a complex constant. To avoid any ambiguity, FORTRAN 5 prohibits redundant parentheses in input and output lists.

The Conversion Aid removes redundant parentheses in input/output lists. For example, the statements:

```
COMPLEX C
PRINT *,(1.,2.),((1.2,2.3),3.4)
PRINT *,(1.1,2.,(2,3))
PRINT *,(2.7*(C+(2.8,10.5)))
```

are converted to the statements:

```
COMPLEX C
PRINT *,(1.,2.),(1.2,2.3),3.4
PRINT *,1.1,2.,2,3
PRINT *,2.7*(C+(2.8,10.5))
```

FORMAT STATEMENTS

FORMAT conversions include mostly minor changes to edit descriptors X, T, and delimited Hollerith constants.

Omitted Commas in FORMAT Statements

FORTRAN 4 permits the commas used to separate list items in a FORMAT statement to be omitted in the following cases: after an X edit descriptor; after an nH edit descriptor; and after a delimited Hollerith edit descriptor. FORTRAN 5 requires that commas separate all list items in a FORMAT statement.

The Conversion Aid replaces any missing commas in a FORMAT statement. The rule for detecting a FORMAT statement is: any statement with a statement label followed by the seven characters FORMAT(is a FORMAT statement. Thus, the statement:

```
10 FORMAT(X5H)=A(N)
```

will be interpreted as a FORMAT statement, even though its syntax is the same as for an arithmetic assignment statement where FORMAT and A are the names of arrays and X5H is the name of a variable.

X Edit Descriptor

FORTRAN 4 allows the n preceding the X edit descriptor to be any positive, negative, or zero integer, or to be omitted entirely. FORTRAN 5 requires that the n be a positive integer. If the n is zero, the Conversion Aid deletes the descriptor. If the n is negative, the Conversion Aid changes it to the new form TLm, where m is the absolute value of n. In all cases, the converted form functions exactly like the old one. For example:

```
50 FORMAT(X,I2,0X,A10,-5X,A3)
```

is converted to:

```
50 FORMAT(1X,I2,A10,TL5,A3)
```

E Edit Descriptor

The FORTRAN 4 floating-point descriptor E can be specified as Ew.dDe with exponent length e. FORTRAN 5 allows only the conversion specification Ew.dEe, but not Ew.dDe. The Conversion Aid changes each Ew.dDe specification into an Ew.dEe specification by changing the D to an E.

Delimited Hollerith Constants

FORTRAN 4 permits character strings delimited by a pair of * or " symbols as alternative forms of the H descriptor. FORTRAN 5 character strings are delimited by apostrophes. An apostrophe within the character string is represented by two consecutive apostrophes. The Conversion Aid changes each constant delimited by * or ". For example:

```
10 FORMAT(1X, "AC""E", *FH*, "J'K")
```

is converted to:

```
10 FORMAT(1X, 'AC"E', 'FH', 'J''K')
```

T Edit Descriptor

FORTRAN 4 allows the FORMAT specification Tn to be used with n=0; T0 is equivalent to T1. FORTRAN 5 requires that the n in a Tn specification be an integer greater than zero. The Conversion Aid changes the specification T0 to T1. If Tn is used in variable formats, the occurrence of T0 is not detected nor is it deleted.

Floating-Point Descriptor

In FORTRAN 4, the d field in the floating-point descriptors Fw.d, Ew.d, Gw.d, and Dw.d, is not required, where:

- d specifies the number of digits to the right of the decimal point within the floating-point field.

In FORTRAN 5, the d field is required in the floating-point descriptors. For example:

```
F8
```

is required to be

```
F8.0
```

If the d field is missing, the Conversion Aid appends an explicit field of zero. For example:

```
F7
```

becomes

```
F7.0
```

PROGRAM UNITS, PROCEDURES, AND OVERLAYS

FORTRAN 5 syntax requirements affect the FORTRAN 4 statements SUBROUTINE (with or without RETURNS), RETURN, CALL, and ENTRY; the sequence for OVERLAY and IDENT has also been changed.

The statement CALL with RETURNS and SUBROUTINE with RETURNS cannot be used in FORTRAN 5. FORTRAN 5 instead requires an asterisk for each statement label dummy argument in the argument list. ENTRY can have a dummy argument. Any reference to an entry point must include an argument list which agrees in order, number, and type with the dummy list in the ENTRY statement.

ALTERNATE RETURN SUBROUTINE LINKAGE

The syntax, but not the function, of the alternate return subroutine linkage differs between FORTRAN 4 and FORTRAN 5.

SUBROUTINE Statement

FORTRAN 4 has dummy alternate return statement labels specified in a separate RETURNS list. FORTRAN 5 has no dummy statement labels. Instead, the programmer includes an asterisk in the dummy argument list for each location in the main program where control could return. The Conversion Aid converts each SUBROUTINE statement from the FORTRAN 4 format to the FORTRAN 5 format. For example:

```
SUBROUTINE AX(A,B), RETURNS(M,N)
```

is converted to:

```
SUBROUTINE AX(A,B,*,*)
```

and:

```
SUBROUTINE AY, RETURNS (P,Q)
```

is converted to:

```
SUBROUTINE AY(*,*)
```

RETURN Statement

For FORTRAN 4, the programmer specifies the dummy alternative return statement label in the RETURN statement. For FORTRAN 5, the programmer specifies a position, rather than a dummy statement label. This position corresponds to the actual argument list of the CALL statement which gives the alternate return statement label. Thus, RETURN 3 specifies that control is to return to the third alternate return statement label specified in the CALL statement. For example:

```
RETURN M
```

where M is the third dummy statement label argument in the SUBROUTINE statement, is converted to:

```
RETURN 3
```

CALL Statement

The actual alternate return statement labels for FORTRAN 4 are specified in a separate RETURNS list. For FORTRAN 5, the actual alternate return statement labels are specified in the same list as the other actual arguments and are preceded by an asterisk. The Conversion Aid converts each CALL statement from the FORTRAN 4 format to the FORTRAN 5 format. For example:

```
CALL AX(R,S), RETURNS(5,10)
```

is converted to:

```
CALL AX(R,S,*5,*10)
```

and:

```
CALL AY, RETURNS(15,20)
```

is converted to:

```
CALL AY(*15,*20)
```

DUMMY ARGUMENTS IN ENTRY STATEMENT

No dummy arguments can appear in FORTRAN 4 statements with ENTRY; instead, the dummy arguments appearing with the FUNCTION statement or SUBROUTINE statement apply to any ENTRY statement within that subprogram. For FORTRAN 5, however, any dummy arguments that apply to an ENTRY point must be specified in that ENTRY statement.

FORTRAN 4 does not recognize a dummy argument list in an ENTRY statement; for each ENTRY statement, the list is the one specified in the subprogram header statement of that program unit. FORTRAN 5 permits a dummy argument list in an ENTRY statement and requires that a reference to an entry point use an actual argument list which agrees in order, number, and type with the dummy argument list in the ENTRY statement. Because FORTRAN 4 entry point references use an actual argument list when parameters are to be passed, a suitable dummy argument list must be added to each ENTRY statement.

The Conversion Aid copies the dummy argument list in the subprogram header statement to each ENTRY statement in that program unit. For example:

```
SUBROUTINE SUB(X,Y,Z)
      .
      .
      .
ENTRY A
      .
      .
      .
ENTRY B
```

is converted to:

```
SUBROUTINE SUB(X,Y,Z)
      .
      .
      .
ENTRY A(X,Y,Z)
      .
      .
      .
ENTRY B(X,Y,Z)
```

OVERLAY DIRECTIVES WITH COMPASS ROUTINES

FORTRAN 4 allows the OVERLAY directive and COMPASS IOENT statement, but the order is reversed and the syntax for the OVERLAY directive is changed in FORTRAN 5. For example:

```
OVERLAY (fname,i,j)

IOENT name
```

is converted to:

```
IOENT   name

LCC     OVERLAY (fname,i,j)
```

SUPPLIED PROCEDURES

FORTRAN 5 has 22 new intrinsic functions; several are for character handling. If the intrinsic function name conflicts with a user-selected name, the conversion program generates an EXTERNAL statement. Other differences in FORTRAN 5 include the random number function, end-of-file processing, and extended memory (ECS/LCM) data transfer.

INTRINSIC FUNCTIONS

FORTRAN 5 provides some new intrinsic functions that are not available in FORTRAN 4 as either intrinsic functions or basic external functions. The names of these new functions might conflict with user-selected names. If the name of any of the new intrinsic functions appears in a context that FORTRAN 5 could interpret as an intrinsic function, and if that name does not appear in an EXTERNAL statement, the Conversion Aid generates an EXTERNAL statement containing that name.

The names of the new intrinsic functions are:

ANINT	ICHAR	LLT
BOOL	IDNINT	LOG
CHAR	INOEX	LOG10
ODIM	LEN	MAX
OINT	LGE	MIN
ONINT	LGT	NEQV
OPROO	LLE	NINT
EQV		

RANDOM NUMBER GENERATOR

The intrinsic function RANF, random number generator, is referenced in FORTRAN 4 as RANF(n). A FORTRAN 5 call to the intrinsic function RANF is RANF(). The Conversion Aid deletes the dummy argument.

END-OF-FILE PROCESSING

When a program compiled by FORTRAN 4 encounters an end-of-file (EOF) during a read, the EOF flag is set. Data items retain their current values, and processing continues with the next executable statement following the READ statement. If the program tries to read from a file when the EOF flag is set, the program terminates with an error message. The EOF function tests for an end-of-file and clears the EOF flag if it is set.

FORTRAN 5 uses an EOF specifier ENO=n, where n is the statement label to be executed at end-of-file. For example:

```
READ(10,100,ENO=500)
```

states that at EOF, execution is to continue from statement label 500.

When a FORTRAN 5 program encounters an EOF during a read, the EOF flag is set; data items are undefined. The program terminates unless the READ statement contains an EOF specifier.

The Conversion Aid adds ENO=n to every READ statement. The ENO=n specifies the next executable statement following the READ statement; if the next executable statement does not have a label, the Conversion Aid generates one with five digits.

TRANSFERRING DATA TO AND FROM ECS/LCM

FORTRAN 4 programs can transfer data to extended memory (ECS/LCM) by either the WRITEC or the MOVLEV utility; data can be transferred from ECS/LCM by either the READEC or the MOVLEV utility. Under FORTRAN 5, the WRITEC and READEC utilities are eliminated, but data can be transferred to and from ECS/LCM by the MOVLEV utility.

The Conversion Aid changes a call to WRITEC or READEC to a call to MOVLEV. The statement CALL WRITEC(s,d,n) is converted to CALL MOVLEV(s,d,n); statement CALL READEC(s,d,n) is converted to CALL MOVLEV(d,s,n). In the argument list, s is a variable or array element specifying the starting address of the data to be transferred; d is a variable or array element specifying the starting address of the receiving location; n is the number of words to be transferred.

DEBUGGING FACILITY

Directives control the FORTRAN 4 debugging facility. FORTRAN 5 can be used with an interactive debugging facility and other debugging options can be selected by FORTRAN 5 control statement parameters. FORTRAN 5 interprets all statements with a C$ in columns 1 and 2 as compiler control directives. Because FORTRAN 5 does not recognize the FORTRAN 4 debugging facility, the C$ debug directives would produce errors during FORTRAN 5 compilation. The Conversion Aid either deletes the $ in column 2 of these directives to create comments, or deletes the entire directive. The programmer selects the option with the OD parameter of the F45 control statement.

LISTING CONTROL DIRECTIVES

Both FORTRAN 4 and FORTRAN 5 allow the programmer to intersperse directives within the program to control the listing. The FORTRAN 5 directives are different in format from the FORTRAN 4 directives. For example:

```
C/  LIST,ALL
```

is changed to:

```
C$  LIST(ALL)
```

and the directive:

```
C/  LIST,NONE
```

is changed to:

```
C$  LIST(ALL=0)
```

If the C/ that precedes the LIST directive in FORTRAN 4 were left unchanged, FORTRAN 5 would treat C/ as a comment rather than a directive.

Certain differences between FORTRAN Extended Version 4 (FORTRAN 4) and FORTRAN Version 5 (FORTRAN 5) require manual changes. If the FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aid Program (Conversion Aid) cannot make a required change, a message is printed for each statement it cannot translate. The Update/Modify listing includes statements flagged for manual changes; these statements are deleted and reinserted, and the lines are listed as replacement lines. These lines can be used as templates to create new lines. If the user ignores the manual change message, the FORTRAN 5 source code might be incorrect. In certain cases, such as DO statements, the new code might compile correctly under FORTRAN 5, but might produce different results during execution.

DO STATEMENTS

The DO statement establishes a controlled loop. Zero trip count and resetting the statement parameters are the two major changes affecting the DO loop for FORTRAN 5.

ZERO TRIP DO LOOPS

Unlike FORTRAN 4, FORTRAN 5 allows a DO loop to be executed zero times. FORTRAN 5 establishes a minimum trip count from the control statement; the minimum trip count is zero. For the statement:

```
DO 20 I= J,K
```

if J is greater than K and if the trip count is zero, the loop is not executed; execution continues after statement 20.

RESETTING DO STATEMENT PARAMETERS

Under FORTRAN 4 neither the control variable nor the indexing parameters of a DO statement should be reset within the range or the extended range of the DO loop; resetting them causes unpredictable results. Under FORTRAN 5, resetting the control variable is prohibited; resetting the indexing parameters does not affect the number of times the DO is executed. The Conversion Aid flags any statement redefining the control variable or indexing parameters in the immediate range of a DO loop, but it does not detect any redefining in the extended range.

LIST-DIRECTED INPUT

A FORTRAN 4 list-directed READ statement reads enough data in an input line to supply values for the variables that are in the iolist. For example, if the input is:

```
15,12,10,11,9,8
```

then a program with the statement:

```
READ*,J,K,L
```

would pick up values 15, 12, and 10. On the next list-directed READ, the values read would be 11, 9, and 8. Under FORTRAN 5, a list-directed READ statement also reads the data in an input line to supply values for the variables in iolist. On the other hand, the next list-directed READ would skip to the next input line. In the example, the values 11, 9, and 8 would be skipped.

LIST-DIRECTED OUTPUT

FORTRAN 4 treats list-directed PRINT statements differently from list-directed WRITE statements.

For FORTRAN 4 list-directed PRINT statements, a blank is output as the first character (carriage control) of each record and as the first character of each line when a long record is continued on another line; the delimiting symbol " is not included with the character output. For list-directed WRITE statements, a blank is output as the first character of each record, but not as the first character of each line when a long record is continued on additional lines; the delimiting symbol " is included with the character output.

FORTRAN 5 treats list-directed PRINT and WRITE statements the same. A blank is output as the first character of each record, and as the first character of each line when a long record is continued on additional lines; the delimiting symbols " (for Hollerith values) or ' (for character values) are not included with the character output.

The Conversion Aid does not translate list-directed output statements, and does not produce any warning messages. The user should verify that the output produced by the translated program, after being compiled by FORTRAN 5, is satisfactory for the particular application.

FORTRAN 4 assumes that an integer variable contains Hollerith data if the variable has a value greater than 248_{-1}. The variable is then output with format A10. FORTRAN 5 does not provide this special interpretation. The user should check that integer variables in output lists do not contain Hollerith data.

FORMAT STATEMENTS

Although the Conversion Aid converts most FORMAT statements, some conversions are not possible. The user must be particularly careful when using edit descriptors for double precision. Other FORMAT difficulties can be corrected by using variables of type CHARACTER.

DOUBLE PRECISION ITEMS IN INPUT/OUTPUT LISTS

FORTRAN 4 permits a double precision input/output list item to correspond to two edit descriptors in a FORMAT statement such as 2A10 or 2D20. FORTRAN 5 requires each double precision input or output list item to correspond to exactly one repeatable edit descriptor; the Conversion Aid does not detect the use of two edit descriptors to describe one double precision input or

output list item. The user must ensure that input and output of double precision items follow the FORTRAN 5 format. For example:

```
10 FORMAT(2A10,2O20)
```

should be changed to:

```
10 FORMAT(A20,O40)
```

H INPUT FORMAT SPECIFICATION

The H specification, under FORTRAN 4, can be used in a FORMAT statement to read Hollerith characters into an existing H field. FORTRAN 5 prohibits the H specification on input, but the user can achieve similar results with a variable of type CHARACTER. The Conversion Aid flags any misused H edit descriptors. If a programmer wanted the input:

```
bTHIS IS A VARIABLE HEADING
```

to be written as:

```
THIS IS A VARIABLE HEADING
```

then the FORTRAN 4 statements:

```
      READ(2,10)
10 FORMAT (27Hbbbbbbbbbbbbbbbbbbbbbbbbbbb)
      WRITE(6,10)
```

would produce the result (b indicates blank). After the read, the FORMAT statement contains the input record. Then a subsequent write produces the correct result. To work properly under FORTRAN 5, the source program would have to be converted to:

```
      CHARACTER X*27
      READ(2,10)X
10    FORMAT (A27)
      WRITE(6,10)X
```

After the READ, the character variable X contains the input record.

V DESCRIPTOR AND EQUALS SIGN

A FORTRAN 4 program can have V descriptors in FORMAT statements to vary the conversion specification, and can have equals signs to vary the integer value used in the conversion. For FORTRAN 5, the programmer must use character data; the V descriptor and equals sign are prohibited. The Conversion Aid flags any FORMAT statement containing the prohibited forms.

EXECUTION TIME FORMAT SPECIFICATION

Under FORTRAN 4, variable FORMAT specifications can be stored in an array, a simple variable, or an array element. FORTRAN 5 allows variable FORMAT specifications to be stored in an array or an array element, but not in a simple variable. The Conversion Aid flags any input or output statement that uses a simple variable as a FORMAT specification.

FUNCTIONS AND SUBROUTINES

The Conversion Aid flags certain uses of functions and subroutines. For example, statement functions and intrinsic functions might be flagged because of possible type mismatch. Some library subroutines, such as sense lights, are no longer available; other subroutines, such as TIME and DATE, can only be referenced as functions. The IOCHECK function is still valid, but can be replaced with parameters in the input statements.

REFERENCING A STATEMENT FUNCTION

When executing a reference to a statement function, FORTRAN 4 replaces each dummy argument with the corresponding actual argument expression. It does not check for possible side effects of an external function reference in the expression, and does not check whether the actual argument has the same type as the dummy argument. FORTRAN 5 requires that each actual argument expression be evaluated and converted in type before substitution. The Conversion Aid does not check statement function references for argument type matching or for side effects.

INTRINSIC FUNCTIONS

FORTRAN 4 allows any intrinsic function referenced in an EXTERNAL statement to be passed as an actual argument if it is not referenced in the program unit. For example:

```
PROGRAM ALPHA(INPUT,OUTPUT)
EXTERNAL AND
    .
    .
    .
BOOL=AND(A,B)
```

The use of intrinsic function AND is not possible in program ALPHA.

FORTRAN 5 allows an intrinsic function referenced in an EXTERNAL statement to be passed as an actual argument, except for two classes of functions: those with a variable number of arguments and those that cause a type conversion. The intrinsic functions which cannot be passed as actual arguments under FORTRAN 5 are as follows:

AMAX0	DMAX1	MAX1
AMAX1	DMIN1	MIN0
AMIN0	FLOAT	MIN1
AMIN1	IDINT	OR
AND	IFIX	REAL
CMPLX	INT	SNGL
DBLE	MAX0	XOR

The Conversion Aid flags any statement that uses a name of an intrinsic function as an actual argument. The user can then change the reference if necessary.

SENSE LIGHTS

FORTRAN 5 does not support the sense light subroutines, SLITE and SLITET. Logical variables can replace the subroutines.

TIME AND DATE FUNCTIONS

FORTRAN 4 permits the operating system routines DATE, JDATE, SECOND, TIME, and CLOCK to be referenced either as functions or as subroutines. Under FORTRAN 4, the functions DATE, SECOND, TIME, and CLOCK are of type REAL; the function JDATE is of type INTEGER.

FORTRAN 5 permits these routines to be referenced as functions but not as subroutines. The functions DATE, JDATE, TIME, and CLOCK are of type CHARACTER; the function SECOND is of type REAL. These functions have no arguments. The Conversion Aid flags any direct reference to these routines; indirect references are not detected.

IOCHEC FUNCTION

Under FORTRAN 4, whenever an error occurs during a read, the parity error flag is set and execution continues with the statement following the READ statement. The IOCHEC function checks for a parity error, then clears the parity flag.

FORTRAN 5 uses the specifiers IOSTAT and ERR for error processing. The format for the I/O status specifier is:

```
IOSTAT=var
```

where the integer var is returned after the read. If var is positive, then an error occurred. The format for the error specifier is:

```
ERR=n
```

where n is a statement label; processing continues at statement n if an error occurs.

When an error occurs during a read, FORTRAN 5 checks for an I/O status specifier or error specifier in the statement; if neither is there, the program terminates. The following examples show FORTRAN 5 READ statements that handle error detection:

```
READ(10,100,IOSTAT=IVAR)

READ(10,100,ERR=500)

READ(10,100,IOSTAT=IVAR,ERR=500)
```

If an error occurs during a read and the READ statement contains IOSTAT, the variable IVAR is set to a positive value indicating the error number. Processing continues with the statement following READ (if there is no ERR) or statement label 500 (if there is ERR=500). If there is no IOSTAT, processing continues at statement label 500.

If a FORTRAN 4 program tests for a parity error with the IOCHEC function, the Conversion Aid issues a warning message. The user can then add an IOSTAT or ERR specifier to the corresponding READ statement. Because FORTRAN 5 does not set the parity error flag, the IOCHEC function works the same as under FORTRAN 4; it clears the parity error flag, and returns a nonzero value if an error has occurred.

NUMERIC CONVERSION OF BLANKS IN FORMATTED INPUT

Embedded blanks in formatted input data items read with numeric conversion are treated differently by FORTRAN 4 and FORTRAN 5. FORTRAN 4 treats blanks embedded in formatted input data as zeros; FORTRAN 5 treats them as null characters. The data item 1bb1, read under an I4 conversion would be read as 1001 by FORTRAN 4 and 11 by FORTRAN 5.

There are two ways to make FORTRAN 5 behavior the same as FORTRAN 4:

1. Put BZ in the format statements in the source program. For example:

   ```
   10 FORMAT(BZ,I10,F10.5,I10)
   ```

2. Add an OPEN statement for the unit prior to the first I/O usage of the unit which specifies BLANK='ZERO'. For example:

   ```
   OPEN(5,BLANK="ZERO')
   ```

Option 1 cannot be used with variable formats unless the variable format is modified. Option 2 cannot be used if the unit is used for both formatted and unformatted I/O because the OPEN statement defaults to formatted.

Control Data operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation is specified when the operating system is installed.

Depending on another installation option, NOS and NOS/BE assume an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use). Under NOS/BE, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The specified mode remains in effect throughout the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card. Under NOS, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card, as described above for a 7/8/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1; 029 mode can be specified by a card with 5/7/9 multipunched in column 1 and a 9 punched in column 2.

Under SCOPE 2 the alternate modes are 026, 029, and blank coded, check next card. The 026 and 029 modes are specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The blank coded, check next card mode is specified according to the following alternatives:

If the next card is a free form flag card, the section following is free form binary. (See SCOPE 2 Reference Manual.)

If the next card has 7/9 punched (only) in column 1, the following section is SCOPE 2 binary. (See SCOPE 2 Reference Manual.)

In any other case the section following is coded with the last requested conversion mode.

When the 63-character set is used, the display code 00 under A or R FORMAT conversion is converted to a space (display code 55_8) for ENCODE and DECODE as well as formatted input/output statements. No conversions occur with A or R FORMAT conversion when the 64-character set is used.

Graphic character representation on a terminal or printer depends on the installation character set and the terminal type. Characters shown as CDC graphic characters in the standard character set table are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

FORTRAN and standard character sets are shown in table A-1.

TABLE A-1. FORTRAN AND STANDARD CHARACTER SETS

FORTRAN	Display Code (octal)	CDC			ASCII		
		Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
: (colon)	00†	: (colon)††	B-2	00	: (colon)††	B-2	072
A	01	A	12-1	61	A	12-1	101
B	02	B	12-2	62	B	12-2	102
C	03	C	12-3	63	C	12-3	103
D	04	D	12-4	64	D	12-4	104
E	05	E	12-5	65	E	12-5	105
F	06	F	12-6	66	F	12-6	106
G	07	G	12-7	67	G	12-7	107
H	10	H	12-8	70	H	12-8	110
I	11	I	12-9	71	I	12-9	111
J	12	J	11-1	41	J	11-1	112
K	13	K	11-2	42	K	11-2	113
L	14	L	11-3	43	L	11-3	114
M	15	M	11-4	44	M	11-4	115
N	16	N	11-5	45	N	11-5	116
O	17	O	11-6	46	O	11-6	117
P	20	P	11-7	47	P	11-7	120
Q	21	Q	11-8	50	Q	11-8	121
R	22	R	11-9	51	R	11-9	122
S	23	S	0-2	22	S	0-2	123
T	24	T	0-3	23	T	0-3	124
U	25	U	0-4	24	U	0-4	125
V	26	V	0-5	25	V	0-5	126
W	27	W	0-6	26	W	0-6	127
X	30	X	0-7	27	X	0-7	130
Y	31	Y	0-8	30	Y	0-8	131
Z	32	Z	0-9	31	Z	0-9	132
0	33	0	0	12	0	0	060
1	34	1	1	01	1	1	061
2	35	2	2	02	2	2	062
3	36	3	3	03	3	3	063
4	37	4	4	04	4	4	064
5	40	5	5	05	5	5	065
6	41	6	6	06	6	6	066
7	42	7	7	07	7	7	067
8	43	8	8	10	8	8	070
9	44	9	9	11	9	9	071
+ (plus)	45	+	12	60	+	12-8-6	053
- (minus)	46	-	11	40	-	11	055
* (asterisk)	47	*	11-8-4	54	*	11-8-4	052
/ (slash)	50	/	0-1	21	/	0-1	057
((left paren)	51	(	0-8-4	34	(	12-8-5	050
) (right paren)	52	)	12-8-4	74	)	11-8-5	051
$ (currency)	53	$	11-8-3	53	$	11-8-3	044
= (equals)	54	=	8-3	13	=	8-6	075
blank	55	blank	no punch	20	blank	no punch	040
, (comma)	56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
. (decimal point)	57	. (period)	12-8-3	73	. (period)	12-8-3	056
	60	≡	0-8-6	36	#	8-3	043
	61	[	8-7	17	[	12-8-2	133
	62	]	0-8-2	32	]	11-8-2	135
	63	%††	8-6	16	%††	0-8-4	045
" (quote)	64	≠	8-4	14	" (quote)	8-7	042
	65	←	0-8-5	35	_ (underline)	0-8-5	137
	66	∨	11-0 or 11-8-2†††	52	!	12-8-7 or 11-0†††	041
	67	∧	0-8-7	37	&	12	046
' (apostrophe)††††	70	↑	11-8-5	55	' (apostrophe)	8-5	047
	71	↓	11-8-6	56	?	0-8-7	077
	72	<	12-0 or 12-8-2†††	72	<	12-8-4 or 12-0†††	074
	73	>	11-8-7	57	>	0-8-6	076
	74	≤	8-5	15	@	8-4	100
	75	≥	12-8-5	75	\	0-8-2	134
	76	¬	12-8-6	76	~ (circumflex)	11-8-7	136
	77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end-of-record mark rather than two colons.
††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55$_8$).
†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.
††††FORTRAN 5 character only.

MESSAGES AND DIAGNOSTICS

B

The Conversion Aid produces a series of messages and diagnostics. Most of the messages are informative. Some messages flag potentially troublesome statements; the user is not obligated to change the flagged statements, but if the changes are not made, the program might execute incorrectly. The error messages (diagnostics) usually require some user action.

Table B-1 contains the automatic conversion (A type) messages. These messages are generated whenever the conversion aid adds, deletes, or replaces a statement. These messages are informative and require no manual action.

Table B-2 contains the manual change (M type) messages. These messages are generated whenever the conversion aid encounters a statement that it cannot translate properly or a statement that might not compile or execute correctly. Often the message is a warning to the user; the statement might be correct, but the user must decide. Some messages indicate flagged statements that must be changed.

Table B-3 contains the error (E type) messages. These messages are generated whenever the conversion aid encounters an error such as a syntax error or an invalid input file. If the conversion aid generates any manual action message, it also generates an error message; this error message is not fatal.

Table B-4 contains messages that are generated when the input stream is being scanned by the Conversion Aid. The messages inform the user of problems in the input stream.

Table B-5 contains dayfile messages.

Table B-6 contains messages that are generated by the Conversion Aid's parser. These messages constitute conditions that are catastrophic to the Conversion Aid and require that the site analyst be consulted.

TABLE B-1. AUTOMATIC CONVERSION MESSAGES

Number	Message	Number	Message
A01	ADDITIONAL MESSAGES SUPPRESSED	***A16***	NEGATIVE OCTAL CONVERTED TO BOOLEAN OCTAL
A04	STATEMENT IGNORED	***A17***	NEGATIVE HOLLERITH CONVERTED TO BOOLEAN
A05	CALL TO -READEC- OR -WRITEC- CHANGED TO -MOVLEV-	***A18***	NEGATED SHIFT CONVERTED TO BOOLEAN SHIFT
A06	HOLLERITH CONSTANT TRUNCATED TO 10 CHARACTERS	***A19***	IRREGULAR INITIALIZATION IN DATA STATEMENT
A07	CONTINUE GENERATED FOR F45 INVENTED LABEL	***A20***	IRREGULAR INITIALIZATION OF DOUBLE PRECISION VARIABLE/ARRAY IN DATA STATEMENT
A08	CONTINUE GENERATED FOR LABELED END STATEMENT	***A21***	UNSUBCRIPTED ARRAY NAME GIVEN SUBSCRIPTS
A09	F45 GENERATED STATEMENT (VARIABLES REPLACE HOLLERITH LITERAL CONSTANTS)	***A22***	PARENTHESES INSERTED TO MAINTAIN ORDER OF SUCCESSIVE EXPONENTIATIONS
A10	LONG HOLLERITH CONSTANT SPLIT INTO 10 CHARACTER PIECES	***A23***	ALTERNATE FORM OF DATA STATEMENT CONVERTED TO ANSI FORM
A11	ACTUAL ARGUMENT USED CONFLICTS WITH FTN5 INTRINSIC FUNCTION NAME	***A24***	NON-ANSI FORMS OF DATA CONSTANT LIST CONVERTED TO ANSI FORM
A12	COMMENT SPECIFIED BY $ CONVERTED TO * OR C	***A25***	KEYWORD TYPE DELETED
A13	MULTIPLE STATEMENTS PER LINE CONVERTED TO ONE STATEMENT PER LINE	***A26***	LEVEL STATEMENT CONVERTED TO CONTAIN COMMON BLOCK NAME
A14	GARBAGE IN COLUMNS 1-5 REPLACED WITH BLANKS	***A27***	DUMMY ARGUMENT LIST APPENDED TO ENTRY STATEMENT
A15	OCTAL NN B CONVERTED TO O=NN=		

TABLE B-1. AUTOMATIC CONVERSION MESSAGES (Contd)

Number	Message	Number	Message
A28	FTN4 - */=/H/ - DELIMITERS CONVERTED TO APOSTROPHES	***A46***	COMPILER LISTING CONTROL DIRECTIVES CONVERTED TO NEW FORM
A29	COMPLEX EXPRESSIONS IN IF CONVERTED TO REAL	***A47***	C$ COMMENT DELETED OR $ PORTION REMOVED PER F45 CONTROL CARD OPTION DD
A30	TWO BRANCH ARITHMETIC IF CONVERTED TO 3 BRANCH IF	***A48***	END STATEMENT INSERTED
A31	ALTERNATE RETURNS IN CALL CONVERTED TO NEW SYNTAX	***A49***	HOLLERITH IN OUTPUT STATEMENT CONVERTED TO APOSTROPHE DELIMITED HOLLERITH
A32	ALTERNATE RETURNS IN SUBPROGRAM CONVERTED TO NEW SYNTAX	***A50***	ABBREVIATIONS OF OPERATORS EXPANDED TO FULL OPERATORS
A33	ALTERNATE RETURNS IN RETURN CONVERTED TO NEW SYNTAX	***A51***	NUMBERED COMMON BLOCK CONVERTED TO LABELED
A34	CALL GOTOER INSERTED AFTER COMPUTED GOTO TO ENSURE AGAINST FAILURE OF GOTO	***A52***	ABBREVIATION DOUBLE EXPANDED TO FULL KEYWORD
A35	LOGICAL IF AND COMPUTED GOTO CONVERTED TO IF-THEN-ELSE FORM	***A53***	REDUNDANT FORM OF OUTPUT STATEMENT CONVERTED TO ANSI FORM
A36	CONTINUED END CONVERTED TO END ON ONE LINE	***A54***	EXPONENT LENGTH SPECIFIER D IN E DESCRIPTOR CONVERTED TO E
A37	DIGIT -1- PREFIXED TO X EDIT DESCRIPTOR	***A55***	DUMMY ARGUMENT DELETED
A38	SPECIFICATION OX DELETED FROM FORMAT	***A56***	ELEMENT OF LEVEL NOT IN COMMON NOR AN FP - DELETED
A39	SPECIFICATION -NX CONVERTED TO TLN SPECIFICATION IN FORMAT	***A57***	INPUT STATEMENT SYNTAX CONVERTED TO ANSI FORM
A40	REDUNDANT PARENTHESES IN I/O LIST DELETED	***A58***	LEVEL DELETED - ELEMENT IN PREVIOUS NEW LEVEL LCC AFTER IDENT
A41	SPECIFICATION TO CONVERTED TO T1	***A59***	OVERLAY BEFORE IDENT CONVERTED TO
A42	FTN4 C$ DEBUG FACILITY NO LONGER PROVIDED	***A60***	EXTERNAL GENERATED TO AVOID FTN4 NAME / FTN5 FUNCTION CONFLICT
A43	PUNCTUATION PLACED AFTER DATA CONVERSION DESCRIPTORS	***A61***	TEMPORARY NAME SUBSTITUTED FOR OVERSUBSCRIPTED ARRAY
A44	LOGICAL 2-BRANCH IF CONVERTED TO IF() GOTO	***A62***	ZERO-FILLED TYPE HOLLERITH CONSTANTS CONVERTED TO FTN5 FORM
A45	BLANK LINE INTERPRETED AS COMMENT IN FTN5	***A63***	FLOATING-POINT DESCRIPTOR GIVEN FIELD OF ZERO

TABLE B-2. MANUAL CHANGE MESSAGES

Number	Message	Significance	Action
M01	POSSIBLE EXTENDED RANGE DO	As stated.	Check range of DO statement.
M02	DO PARAMETER REDEFINED WITHIN RANGE OF DO	As stated.	Make sure number of times DO loop is executed is not affected.
M03	FUNCTIONS -SLITE- AND -SLITET- NOT SUPPORTED IN FTN5	As stated.	Replace function with logical variable.
M04	-DATE, JDATE, TIME, CLOCK, SECOND- HAVE NO ARGUMENTS IN FTN5	As stated.	Rewrite statements that depend on argument.
M05	SYNTAX ERROR IN TWO BRANCH ARITHMETIC IF STATEMENT	As stated.	Correct IF statement.
M06	SIMPLE VARIABLE USED AS FORMAT SPECIFIER IN I/O STATEMENT	As stated.	Eliminate simple variable.
M07	V OR = DESCRIPTOR USED IN FORMAT STATEMENT. CHECK I/O USING THIS FORMAT	As stated.	Eliminate V and = descriptors.
M09	FORMAT CONTAINING HOLLERITH DESCRIPTOR USED BY INPUT STATEMENT	As stated.	Use character data instead.
M10	NEGATIVE HOLLERITH CONSTANT LARGER THAN 50 CHARS IN DATA STATEMENT	As stated.	Rework Hollerith constant.
M11	FTN4 INTRINSIC FUNCTION USED AS ACTUAL ARGUMENT	As stated.	Check function name. Some intrinsic functions not allowed as actual arguments in FORTRAN 5.
M12	TYPE ECS CONVERTED TO LEVEL 3. CHECK CONVERSION	As stated.	Check conversion.
M13	HOLLERITH PARAMETERS - CHECK DEPENDENCY ON TRAILING ZEROS	As stated.	Check dependency.
M14	TYPE MISMATCH	As stated.	Change type.
M15	IF EXPRESSION NOT TYPED LOGICAL OR NOT ARITHMETIC IF	As stated.	Check validity of IF statement.
M16	SECOND PARAMETER OF READEC NOT IN LEVEL OR IS NUMERIC VALUE	As stated.	Check READEC.
M17	READEC/WRITEC WITHOUT ECS OR LEVEL STATEMENT - NOT CONVERTED	As stated.	Check statement.
M18	CHECK USAGE OF ASF	Arithmetic statement function is incorrect. Syntax is incorrect or use of dummy arguments is not correct for FORTRAN 5.	Check arithmetic statement function and all references. Correct as necessary.
M19	DUMMY ARGUMENT REFERENCE MUST BE SIMPLE VARIABLE	As stated.	Change dummy argument to a simple variable.
M20	NO CONVERSION OF COMPLEX TO REAL - REAL USED AS VARIABLE	As stated.	Check conversion.

TABLE B-2. MANUAL CHANGE MESSAGES (Contd)

Number	Message	Significance	Action
M21	FUNCTION IOCHEC . . . MANUAL INSPECTION REQUIRED	As stated.	Check IOCHEC function usage.
M22	LIST(C=0) IGNORED DUE TO NEW SOURCE - CHECK UPDATE/MODIFY DIRECTIVE CONFLICTS	As stated.	Check conversion.
M23	MACHINE DEPENDENT CODING IN ABOVE STATEMENT	As stated.	Check the machine-dependent coding.
M24	SUBSCRIPT OF ARRAY EXCEEDS DIMENSION	As stated.	Change subscript.
M25	PARTIAL LIST DIRECTED READ TREATED DIFFERENT BY FTN5	As stated.	Check input to list-directed READ for possible skipping of input values.
M26	IMPLICIT STATEMENT DELETED	FTN4 ignores all but the first IMPLICIT statement. FTN5 accepts multiple IMPLICIT statements. All but the first IMPLICIT statement have been deleted.	Check the deleted IMPLICIT statements.

TABLE B-3. ERROR MESSAGES

Number	Message	Significance	Action
E01	INPUT NOT COMPILE FILE, UPDATE/MODIFY OUTPUT SPECIFIED...NO OUTPUT GENERATED	Input/output file mismatch.	Check input file.
E02	INPUT NOT SEQUENCED AS SPECIFIED ON CONTROL CARD...NO CONVERSION DONE	As stated.	Check input file.
E03	SYNTAX ERROR IN INPUT STREAM	One or more FORTRAN syntax errors detected.	Correct FORTRAN errors in the program.
E04	MANUAL CONVERSION MESSAGES ENCOUNTERED	As stated.	See the manual conversion messages.
E05	FATAL ERROR - NO FURTHER UNIQUE 5 DIGIT LABELS COULD BE GENERATED	As stated.	Rework program to eliminate need of some labels.

TABLE B-4. INPUT SCANNING ERROR MESSAGES

Message	Significance	Action
EXCESS CARDS IN SOURCE STATEMENT	Too many continuation cards.	Correct source statement.
ERROR, SCAN INCOMPLETE	Syntax error in the source statement.	Correct source statement.
STATEMENT LABEL WITH NO STATEMENT	As stated.	Correct source statement.
ABOVE PROGRAM UNIT ALREADY PROCESSED BY F45..WILL BE SKIPPED	The comment C*F45V1P0* was detected prior to the first card.	No action.

TABLE B-4. INPUT SCANNING ERROR MESSAGES (Contd)

Message	Significance	Action
UNRECOGNIZED STATEMENT-- CHECK FOR JCL IN INPUT STREAM	As stated.	Remove JCL from input stream.
FURTHER JCL WILL GO UNDETECTED-- INCORRECT OUTPUT MAY RESULT	Error occurs after 10 occurrences of unrecognized statements.	Remove JCL from input stream.
ERROR AT LINE nn CARD NO. nnn COLUMN kkk SYNTAX ERROR	As stated; where nn = line number, nnn = card number, kkk = column number.	Check source statement for syntax error.

TABLE B-5. DAYFILE MESSAGES

Message	Significance	Action
NULL INPUT	No data in input stream.	Check input file.
INPUT TYPE CONFLICTS WITH OUTPUT DESIRED	Most likely to occur with regular input and with an Update/Modify directive, requested for output.	Check type of input.
INPUT TYPE NOT SAME AS SPECIFIED ON CC	Most likely to occur when specifying sequenced input on control card (CC) and input is not sequenced.	Check type of input.
SYNTAX ERRORS IN INPUT	As stated.	Check input source statements for syntax errors.
NO. OF MANUAL CHANGES REQUIRED = nn	A total of nn manual change messages were output during a Conversion Aid run.	Analyze output source statements for possible manual corrections.

TABLE B-6. PARSING ERROR MESSAGES

Message	Significance	Action
PARSE TERMINATED BY QUALIFICATION STACK OVERFLOW	As stated.	Consult site analyst.
PARSE TERMINATED BY NEST STACK OVERFLOW	As stated.	Consult site analyst.
TRANSLATION STRUCTURE OVERFLOW	As stated.	Consult site analyst.

GLOSSARY C

This glossary does not include terms defined in the ANSI standard for FORTRAN, X3.9-1966 (FORTRAN 66) or FORTRAN, X3.9-1978 (FORTRAN 77).

Blank Common Block -
An unlabeled common block. No data can be stored into a blank common block at load time. The size of the block is determined by the largest declaration for it. Contrast with Labeled Common Block.

Call by Name -
A method of referencing a subprogram in which the addresses of the actual arguments are passed.

Call by Value -
A method of referencing a subprogram in which only the values of the actual arguments are passed.

Common Block -
An area of memory that can be declared in a COMMON statement by more than one relocatable program and used for storage of shared data. See Blank Common Block and Labeled Common Block.

End-of-File (EOF) -
A particular kind of boundary on a sequential file, recognized by the functions EOF and UNIT, and written by the ENDFILE statement. Any of the following conditions is recognized as end-of-file.

End-of-section (for INPUT file only)

End-of-partition

End-of-information (EOI)

W type record with flag bit set and delete bit not set

Tape mark

Trailer label

Embedded zero length level 17_8 block

Entry Point -
A location within a program unit that can be branched to from other program units. Each entry point has a unique name.

External Reference -
A reference in one program unit to an entry point in another program unit.

File -
A logically related set of information; the largest collection of information that can be addressed by a file name. A file starts at beginning-of-information and ends at end-of-information.

Labeled Common Block -
A common block into which data can be stored at load time. The first program unit declaring a labeled common block determines the amount of memory allocated. Contrast with Blank Common Block.

Logical File Name -
The name by which a file is identified; consists of one to seven letters or digits, the first a letter. Files used in FORTRAN input/output statements must be defined in the PROGRAM statement, and can have a maximum of six letters or digits.

Program Unit -
A sequence of FORTRAN statements terminated by an END statement. The FORTRAN program units are main programs, subroutines, functions, and block data subprograms.

Sequential -
A file organization in which the location of each record is defined only as occurring immediately after the preceding record. A file position which specifies the next record to be read or written is defined at all times.

Source Code -
Code written by the programmer in a language such as FORTRAN, and used as input to a compiler.

Source Listing -
A compiler-produced listing, in a particular format, of the original source program.

Time-Sharing Mode -
One of the compilation modes in the FORTRAN Extended compiler, indicated by the TS control statement option.

Unit Designator -
An integer constant or variable with a value 0 through 99, or an L format logical file name. In input/output statements, the unit designator is linked with the actual file name by the PROGRAM statement and indicates on which file the operation is to be performed.

PROGRAM EXAMPLES

D

This appendix provides some examples of programs converted with the Conversion Aid. An example showing standard input is shown in figure D-1. An example showing sequenced input is shown in figure D-2. An example showing COMPILE file input is shown in figure D-3.

```
FORTRAN 4 TO 5                    FORTRAN CONVERSION PROGRAM      F4S 1.0          01/16/79  12.21.29.       PAGE      1

                      PROGRAM DEV1L29 (TAPE6)                                                          000100
                C     FIV056 - USE OF SINGLE SUBSCR1PT 1N AN EQUIVALENCE STATEMENT                     000110
                C            WITH ARRAYS WH1CH HAVE TWO AND THREE D1MENS1ONS AND                      000120
                C            WHERE THE ARRAYS ARE OF THE SAME AND D1FFERENT TYPE5.                    000130
           5    C     TEST ASSUMES THAT DP QUANT1TY TAKE5 EXACTLY TWICE THE SPACE                      000140
                C     OF A REAL QUANT1TY                                                               000150
                      INTEGER P                                                                        000160
                      1NTEGER A(40), B(4,5)                                                            000170
                      D1MENS1ON C(3,4,5), F(40)                                                        000180
          10          DOUBLE PRECISION D(5,4), E(5,5,3 )                                               000190
                      EQU1VALENCE ( B(20), A(10) ), ( A(40), C(20) ),  (F,C(21) ),                     000200
                     1 ( C(59), D(20) ), ( D(3), E(53)  )                                              000210
                      P=6                                                                              000220
                      DO 1 J=1,5                                                                       000230
          15          DO 1 I=1,4                                                                       000240
                1     B(I,J)= I + 4*(J-1)                                                              000250
                      WR1TE( P,2 )                                                                     000260
                2     FORMAT( 7H1F1V056 / 53H THE FOLLOWING PAIR5 OF SEQUENCE5 SHOULD BE               000270
                     1 1DENT1CAL   //  )                                                              000280
          20          WR1TE( P,3) B(3,3), B(4,3),  ( ( B(1,J),I=1,4), J=4,S ),                         000290
                     1 ( A(I),1=1,10 )                                                                 000300
                3     FORMAT( 13H SEQUENCE 1 ≤  / 1H , 1014  /                                         000310
                     1 13H 5EQUENCE 2 ≤  /  1H ,10I4   /  )                                           000320
                      DO 4 K=1,5                                                                       000330
          25          DO  4  J=1,4                                                                     000340
                      DO 4  1=1,3                                                                      000350
                4     C(1,J,K)= 1 + 3*(J-1) + 12*(K-1)                                                 000360
                      WR1TE(P,5 ) C(3,3,2), ( C(1,4,2),I=1,3),                                         000370
                     1 (((C(I,J,K),I=1,3),J=1,4),K=3,5 ), F                                            000380
          30     S     FORMAT( 13H SEQUENCE 1 ≤ / 2( 1H , 20F5.1 / )                                    000390
                     1 13H SEQUENCE 2 ≤ / 2( 1H , 20F5.0 / ) / )                                       000400
                      DO 6 K=1,3                                                                       000410
                      DO 6 J=1,5                                                                       000420
                      DO 6 I=1,5                                                                       000430
          35          E(I,J,K)= I + 5*(J-1) + 25*(K-1)                                                 000440
                6      WRITE( P,7) (( E(I,J,3),1=1,5), J=1,4),                                         000450
                     1 (( D(1,J), I=1,5), J=1,4 )                                                      000460
                7     FORMAT( 13H SEQUENCE 1 ≤  / 2( 1H , 2P10D10.1  /  )                              000470
                     1 13H SEQUENCE 2 ≤  /  2( 1H , 2P10D10.1  /  ))                                   000480
          40          STOP                                                                             000490
                      END                                                                              000500

     NO. OF CARD5=   41        NO. OF ERROR5=  0
```

Figure D-1. Standard Input Example (Sheet 1 of 5)

```
                    DEVIL29          FORTRAN CONVERSION PROGRAM      F45 1.0        01/16/79  12.21.29.      PAGE      1
C*F45V1P0*                                                                                                    -I-
      PROGRAM DEVIL29 (TAPE6)                                                                                 -R-
C     FIV056 - USE OF SINGLE SUBSCRIPT IN AN EQUIVALENCE STATEMENT                    000110
C              WITH ARRAYS WHICH HAVE TWO AND THREE DIMENSIONS AND                    000120
C              WHERE THE ARRAYS ARE OF THE SAME AND DIFFERENT TYPES.                  000130
C     TEST ASSUMES THAT DP QUANTITY TAKES EXACTLY TWICE THE SPACE                     000140
C     OF A REAL QUANTITY                                                              000150
      INTEGER P                                                                       000160
      INTEGER A(40), B(4,5)                                                           000170
      DIMENSION C(3,4,5), F(40)                                                       000180
      DOUBLE PRECISION D(5,4), E(5,5,3 )                                              000190
      INTEGER ZZZ112(20)                                                                                      -I-
      EQUIVALENCE (ZZZ112(1),B(1,1))                                                                          -I-
      REAL ZZZ113(60)                                                                                         -I-
      EQUIVALENCE (ZZZ113(1),C(1,1,1))                                                                        -I-
      DOUBLE PRECISION ZZZ114(20)                                                                             -I-
      EQUIVALENCE (ZZZ114(1),D(1,1))                                                                          -I-
      DOUBLE PRECISION ZZZ115(75)                                                                             -I-
      EQUIVALENCE (ZZZ115(1),E(1,1,1))                                                                        -I-
      EQUIVALENCE ( ZZZ112(20), A(10) ), ( A(40), ZZZ113(20) ),  (F(1),Z                                      -R-
     1ZZ113(21) ),       ( ZZZ113(59), ZZZ114(20) ), ( D(3,1), ZZZ115(53                                     -R-
     1)  )                                                                                                    -R-
          ***A61*** TEMPORARY NAME SUBSTITUTED FOR OVERSUBSCRIPTED ARRAY
          ***A21*** UNSUBCRIPTED  ARRAY NAME GIVEN SUBSCRIPTS
      P=6                                                                 000220
      DO 1 J=1,5                                                          000230
      DO 1 I=1,4                                                          000240
1     B(I,J)= I + 4*(J-1)                                                 000250
      WRITE( P,2 )                                                        000260
2     FORMAT( 7H1F1V056 / 53H THE FOLLOWING PAIRS OF SEQUENCES SHOULD BE
     1 IDENTICAL   //  )                                                 000280
      WRITE( P,3) B(3,3), B(4,3),  ( ( B(I,J),I=1,4), J=4,5 ),
     1 ( A(I),I=1,10 )                                                   000300
3     FORMAT( 13H SEQUENCE 1 =  / 1H , 1014  /
     1 13H SEQUENCE 2 =  /  1H ,1014   /  )                              000320
      DO 4 K=1,5                                                          000330
      DO  4  J=1,4                                                        000340
      DO 4  I=1,3                                                         000350
4     C(I,J,K)= 1 + 3*(J-1) + 12*(K-1)                                    000360
      WRITE(P,5 ) C(3,3,2), ( C(I,4,2),I=1,3),
     1 (((C(I,J,K),I=1,3),J=1,4),K=3,5 ), F                              000380
5     FORMAT( 13H SEQUENCE 1 = / 2( 1H , 20F5.1 / )
     1 13H SEQUENCE 2 = / 2( 1H , 20F5.0 / ) / )                         000400
      DO 6 K=1,3                                                          000410
      DO 6 J=1,5                                                          000420
      DO 6 I=1,5                                                          000430
6     E(I,J,K)= 1 + 5*(J-1) + 25*(K-1)                                    000440
       WRITE( P,7) (( E(1,J,3),I=1,5), J=1,4),
     1 (( D(I,J), I=1,5), J=1,4 )                                        000460
7     FORMAT( 13H SEQUENCE 1 =  / 2( 1H , 2P10D10.1  /  )
     1 13H SEQUENCE 2 =  /  2( 1H , 2P10D10.1  /  ))                     000480
      STOP                                                                000490
      END                                                                 000500
```

Figure D-1. Standard Input Example (Sheet 2 of 5)

```
MESSAGE SUMMARY   DEVIL29          FORTRAN CONVERSION PROGRAM      F45 1.0          01/16/79  12.21.29.      PAGE      2

 CODE  FREQUENCY

 A21         1
 A61         1

-DIAGNOSTICS-

-STATISTICS-

  CARDS         41          CP SECS     2.786          CARDS/MIN       882
```

Figure D-1. Standard Input Example (Sheet 3 of 5)

```
C*F45V1P0*
      PROGRAM DEVIL29 (TAPE6)
C     FIV056 - USE OF SINGLE SUBSCRIPT IN AN EQUIVALENCE STATEMENT       000110
C              WITH ARRAYS WHICH HAVE TWO AND THREE DIMENSIONS AND       000120
C              WHERE THE ARRAYS ARE OF THE SAME AND DIFFERENT TYPES.     000130
C     TEST ASSUMES THAT DP QUANTITY TAKES EXACTLY TWICE THE SPACE        000140
C     OF A REAL QUANTITY                                                 000150
      INTEGER P                                                          000160
      INTEGER A(40), B(4,5)                                              000170
      DIMENSION C(3,4,5), F(40)                                          000180
      DOUBLE PRECISION D(5,4), E(5,5,3 )                                 000190
      INTEGER ZZZI12(20)
      EQUIVALENCE (ZZZI12(1),B(1,I))
      REAL ZZZI13(60)
      EQUIVALENCE (ZZZI13(1),C(1,1,1))
      DOUBLE PRECISION ZZZI14(20)
      EQUIVALENCE (ZZZI14(I),D(1,1))
      DOUBLE PRECISION ZZZI15(75)
      EQUIVALENCE (ZZZI15(I),E(I,1,I))
      EQUIVALENCE ( ZZZI12(20), A(10) ), ( A(40), ZZZI13(20) ),  (F(1),Z
     1ZZI13(21) ),       ( ZZZI13(59), ZZZI14(20) ), ( D(3,1), ZZZI15(53
     I)  )
      P=6                                                                000220
      DO 1 J=I,5                                                         000230
      DO I I=I,4                                                         000240
I     B(I,J)= I + 4*(J-1)                                                000250
      WRITE( P,2 )                                                       000260
2     FORMAT( 7H1FIV056 / 53H THE FOLLOWING PAIRS OF SEQUENCES SHOULD BE
     1 IDENTICAL    //  )                                               000280
      WRITE( P,3) B(3,3), B(4,3),  ( ( B(I,J),I=1,4), J=4,5 ),
     1 ( A(I),I=I,I0 )                                                  000300
3     FORMAT( I3H SEQUENCE 1 S  / 1H , I0I4  /
     1 13H SEQUENCE 2 S  /  1H ,I0I4   /  )                             000320
      DO 4 K=1,5                                                         000330
      DO  4  J=1,4                                                       000340
      DO 4  I=1,3                                                        000350
4     C(I,J,K)= I + 3*(J-1) + 12*(K-1)                                   000360
      WRITE(P,5 ) C(3,3,2), ( C(I,4,2),I=I,3),
     1 (((C(I,J,K),I=1,3),J=1,4),K=3,5 ), F                             000380
5     FORMAT( 13H SEQUENCE 1 S / 2( 1H , 20F5.1 / )
     1 13H SEQUENCE 2 S / 2( 1H , 20F5.0 / ) / )                        000400
      DO 6 K=1,3                                                         000410
      DO 6 J=I,5                                                         000420
      DO 6 I=1,5                                                         000430
6     E(I,J,K)= I + 5*(J-I) + 25*(K-I)                                   000440
       WRITE( P,7) (( E(I,J,3),I=1,5), J=I,4),
     1 (( D(I,J), I=1,5), J=I,4 )                                       000460
7     FORMAT( 13H SEQUENCE 1 S  / 2( IH , 2P10D10.I  /  )
     1 13H SEQUENCE 2 S  /  2( 1H , 2P10D10.1  /  ))                    000480
      STOP                                                               000490
      END                                                                000500
```

Figure D-1. Standard Input Example (Sheet 4 of 5)

```
  MFS  NB1-  CYB74-SN108    5C/R0B    11/14/78
12.19.42.EHFCPHN  FROM     /GS
12.19.42.IP  00000128 WORDS - FILE INPUT  , DC 04
12.19.42.EHFCP (T300, P1, EC1)       0186, 7182, 4
12.19.42.3LM1040, FTNERH, X7333, SVL166.
12.19.44.ATTACH,FTNLIB,ID=CPD.
12.19.44.PFN IS
12.19.44.FTNLIB
12.19.45.PF CYCLE NO. = 007
12.19.45.LIBRARY(FTNLIB)
12.19.46.REWIND,OUTPUT.
12.19.47.ATTACH(DUP2,ID=FTNERH)
12.19.51.  DUP2    CY = 1
12.19.51.F45,I=DUP2,LO=F,P=POUT,PO=F.
12.21.37.    STOP  F45
12.21.37.REWIND,POUT.
12.21.38.COPYSBF(POUT,OUTPUT)
12.21.38. END OF INFORMATION ENCOUNTERED.
12.21.39.EXIT(S)
12.21.39.OP  00001472 WORDS - FILE OUTPUT , DC 40
12.21.39.MS    3584  WORDS  (   25088 MAX USED)
12.21.39.CPA       3.228 SEC.         3.228 ADJ.
12.21.39.CPB        .247 SEC.          .247 ADJ.
12.21.39.IO        1.242 SEC.         1.242 ADJ.
12.21.39.CM       98.930 KWS.         6.038 ADJ.
12.21.39.EC        2.411 KWS.          .073 ADJ.
12.21.39.SS                           10.829
12.21.39.PP        7.940 SEC.     DATE 01/16/79
12.21.39.EJ  END OF JOB, GS
```

Figure D-1. Standard Input Example (Sheet 5 of 5)

```
FORTRAN 4 TO 5                    FORTRAN CONVERSION PROGRAM      F45 1.0      01/16/79  12.33.07.      PAGE      1.

                  00100 PROGRAM L60(OUTPUT)
                  00200*C
                  00300*C  NONSTANOARO INDEXING.  ARRAY SUBSCRIPT IS ANOTHER SUBCRIPTED AR
                  00400*C  RAY NAME.
            5     00500*C
                  00600 INTEGER II,N,M,L,STD
                  00700 OIMENSION II(5,5),N(5,5),M(43),L(43),STD(5,5)
                  00800 COMMON LFLAG
                  00900 OATA ((( II(I,J),I=1,5),J=I,5)=25*(0)),(((N(I,J),I=I,5),J=I,5)=II,
           10     0I000+2I,3I,41,0,I2,22,32,42,0,I3,23,33,43,1I*(0))
                  0II00 DATA ((L(I),I=I,43)=I0*(0),I,2,3,7(0),I,2,3,7*(0),I,2,3,7(0),1,2,3
                  0I200+)
                  0I300 LFLAG=0
                  0I400 00 I00 I=I,43
           15     0I500 I00 M(I)=I
                  0I600 DO I I=I,4
                  01700 DO I J=1,3
                  01800*C  M ANO N ARE ALSO SUBSCRIPTEO ARRAYS.
                  01900 I II(L(M(N(I,J)))),I)=I0*I+J
           20     02000 00 2 I=I,5
                  02I00 DO 2 J=I,5
                  02200 2 STD(I,J)=N(J,I)
                  02300 CALL COMPAR2(II,3H II,STD,5,5)
                  02400 IF(LFLAG.NE.0) GO TO 99
           25     02500 PRINT 20
                  02600 20 FORMAT(* NONSTANDARD INDEXING WITH NESTEO ARRAY SUBSCRIPTS IS O
                  02700+K*)
                  02800 99 CONTINUE
                  02900 END
           30     03000 SUBROUTINE COMPAR2(ARRAY,NAME,STD,II,JJ)
                  03I00 INTEGER ARRAY,STD
                  03200 OIMENSION ARRAY(II,JJ),STO(II,JJ)
                  03300 COMMON LFLAG
                  03400 DO 2 I=1,II
           35     03500 00 2 J=I,JJ
                  03600 IF(ARRAY(I,J).EQ.STD(I,J)) GO TO 2
                  03700 LFLAG=I
                  03800 CALL QAQAQF
                  03900 PRINT I2,NAME,I,J,ARRAY(I,J),STD(I,J)
           40     04000 I2 FORMAT(* ERROR,*,A7,*(*,I3,I3,*)=*,I9  ,* SHOULD BE *,I9  )
                  04I00 2 CONTINUE
                  04200 RETURN
                  04300 ENO

      NO. OF CARDS=   43        NO. OF ERRORS=  0
```

Figure D-2. Sequenced Input Example (Sheet 1 of 6)

```
                    L60                FORTRAN CONVERSION PROGRAM      F45 I.0          0/16/79  I2.33.07.       PAGE      I

100C*F45VIP0*                                                                                                    -I-
110 PROGRAM LGO(OUTPUT)                                                                                          -R-
I20*C
I30*C  NONSTANDARD INDEXING.  ARRAY SUBSCRIPT IS ANOTHER SUBCRIPTED AR
140*C  RAY NAME.
150*C
160 INTEGER II,N,M,L,STD
170 DIMENSION II(5,5),N(5,5),M(43),L(43),STD(5,5)
180 COMMON LFLAG
190 DATA (( II(I,J),I=1,5),J=I,5)/25*0/,((N(I,J),I=I,5),J=1,5)/11,                                               -R-
200+  21,31,41,0,12,22,32,42,0,13,23,33,43,11*0/                                                                -R-
          ***A23*** ALTERNATE FORM OF DATA STATEMENT CONVERTED TO ANSI FORM
          ***A24*** NON-ANSI FORMS OF DATA CONSTANT LIST,CONVERTED TO ANSI FORM
210 DATA (L(I),I=I,43)/10*0,1,2,3,7*0,1,2,3,7*0,1,2,3,7*0,1,2,3       /                                          -R-
          ***A23*** ALTERNATE FORM OF DATA STATEMENT CONVERTED TO ANSI FORM
          ***A24*** NON-ANSI FORMS OF DATA CONSTANT LIST,CONVERTED TO ANSI FORM
220 LFLAG=0
230 DO 100 I=I,43
240 100   M(I)=I
250 DO 1 I=1,4
260 DO 1 J=1,3
270*C  M AND N ARE ALSO SUBSCRIPTED ARRAYS.
280 1     II(L(M(N(I,J))),I)=10*I+J
290 DO 2 I=1,5
300 DO 2 J=1,5
310 2     STD(I,J)=N(J,I)
320 CALL COMPAR2(II,↑ II↑,STD,5,5)                                                                               -R-
          ***A28*** FTN4 - */≠/H - DELIMITERS CONVERTED TO APOSTROPHES
          ***MI3*** HOLLERITH PARAMETERS - CHECK DEPENDENCY ON TRAILING ZEROS
330 IF(LFLAG.NE.0) GO TO 99
340 PRINT 20
350 20    FORMAT(↑ NONSTANDARD INDEXING WITH NESTED ARRAY SUBSCRIPTS IS                                          -R-
360+0        K↑)                                                                                                 -R-
          ***A28*** FTN4 - */≠/H - DELIMITERS CONVERTED TO APOSTROPHES
370 99    CONTINUE
380 END
```

Figure D-2. Sequenced Input Example (Sheet 2 of 6)

```
MESSAGE SUMMARY   L60              FORTRAN CONVERSION PROGRAM      F45 1.0        01/16/79  12.33.07.      PAGE      2

 CODE  FREQUENCY

 A23         2
 A24         2
 A28         2
 M13         1
```

```
                    COMPAR2            FORTRAN CONVERSION PROGRAM      F45 1.0
100C*F45V1P0*                                                                                          -I-
110 SUBROUTINE COMPAR2(ARRAY,NAME,STD,II,JJ)                                                           -R-
120 INTEGER ARRAY,STD
130 DIMENSION ARRAY(II,JJ),STD(II,JJ)
140 COMMON LFLAG
150 DO 2 I=1,II
160 DO 2 J=1,JJ
170 IF(ARRAY(I,J).EQ.STD(I,J)) GO TO 2
180 LFLAG=1
190 CALL QAQAQF
200 PRINT 12,NAME,I,J,ARRAY(I,J),STD(I,J)
210 12    FORMAT(' ERROR,',A7,'(',I3,I3,')=',I9  ,' SHOULD BE ',I9  )                                  -R-
          ***A28*** FTN4 - */*/H - DELIMITERS CONVERTED TO APOSTROPHES
220 2     CONTINUE
230 RETURN
240 END
```

Figure D-2. Sequenced Input Example (Sheet 3 of 6)

```
MESSAGE SUMMARY   COMPAR2          FORTRAN CONVERSION PROGRAM      F45 1.0        01/16/79  12.33.07.      PAGE      2

 CODE  FREQUENCY
 A28         1

-DIAGNOSTICS-
        ***E04*** MANUAL CONVERSION MESSAGES ENCOUNTERED

-STATISTICS-

  CARDS         43          CP SECS      1.846          CARDS/MIN      1397
```

Figure D-2. Sequenced Input Example (Sheet 4 of 6)

```
100C*F45V1P0*
110 PROGRAM LGO(OUTPUT)
120*C
130*C  NONSTANDARD INDEXING.  ARRAY SUBSCRIPT IS ANOTHER SUBCRIPTED AR
140*C  RAY NAME.
150*C
160 INTEGER I1,N,M,L,STD
170 DIMENSION I1(5,5),N(5,5),M(43),L(43),STD(5,5)
180 COMMON LFLAG
190 DATA (( I1(I,J),I=1,5),J=1,5)/25*0/,((N(I,J),I=1,5),J=1,5)/11,
200+  21,31,41,0,12,22,32,42,0,13,23,33,43,11*0/
210 DATA (L(I),I=1,43)/10*0,1,2,3,7*0,1,2,3,7*0,1,2,3,7*0,1,2,3        /
220 LFLAG=0
230 DO 100 I=1,43
240 100   M(I)=I
250 DO 1 I=1,4
260 DO 1 J=1,3
270*C  M AND N ARE ALSO SUBSCRIPTED ARRAYS.
280 1     II(L(M(N(I,J))),1)=10*I+J
290 DO 2 I=1,5
300 DO 2 J=1,5
310 2     STD(I,J)=N(J,I)
320 CALL COMPAR2(II,' II',STD,5,5)
330 IF(LFLAG.NE.0) GO TO 99
340 PRINT 20
350 20    FORMAT(' NONSTANDARD INDEXING WITH NESTED ARRAY SUBSCRIPTS IS
360+0        K')
370 99    CONTINUE
380 END
100C*F45V1P0*
110 SUBROUTINE COMPAR2(ARRAY,NAME,STD,II,JJ)
120 INTEGER ARRAY,STD
130 DIMENSION ARRAY(II,JJ),STD(II,JJ)
140 COMMON LFLAG
150 DO 2 I=1,II
160 DO 2 J=1,JJ
170 IF(ARRAY(I,J).EQ.STD(I,J)) GO TO 2
180 LFLAG=1
190 CALL QAQAQF
200 PRINT 12,NAME,I,J,ARRAY(I,J),STD(I,J)
210 12    FORMAT(' ERROR,',A7,'(',I3,I3,')=',I9  ,' SHOULD BE ',I9  )
220 2     CONTINUE
230 RETURN
240 END
```

Figure D-2. Sequenced Input Example (Sheet 5 of 6)

```
  MFS  NB1-  CYB74-SN108    5C/R0B    11/14/78
12.32.55.EHFCP1T  FROM    /GS
12.32.55.1P  00000128 WORDS - FILE INPUT  , DC 04
12.32.55.EHFCP (T300, P1, EC1)      018G, 7182, 4
12.32.55.3LM1040, FTNERH, X7333, SVL166.
12.32.59.ATTACH,FTNLIB,ID=CPD.
12.32.59.PFN IS
12.32.59.FTNLIB
12.33.00.PF CYCLE NO. = 007
12.33.00.LIBRARY(FTNLIB)
12.33.00.REWIND,OUTPUT.
12.33.01.ATTACH(DUP3,ID=FTNERH)
12.33.02.  DUP3    CY = 3
12.33.03.F45,I=DUP3,LO=F,P=POUT,PO=F,S1,SO=100/10
12.33.03./3.
12.33.14.NO. OF MANUAL CHANGES REQUIRED =
12.33.14.  0000000000000001
12.33.14.    STOP  F45
12.33.14.REWIND,POUT.
12.33.15.COPYSBF(POUT,OUTPUT)
12.33.16. END OF INFORMATION ENCOUNTERED.
12.33.16.EXIT(S)
12.33.16.OP  00000960 WORDS - FILE OUTPUT , DC 40
12.33.16.MS    3584  WORDS  (   25088 MAX USED)
12.33.16.CPA       2.276 SEC.        2.276 ADJ.
12.33.16.CPB        .286 SEC.         .286 ADJ.
12.33.16.IO        1.225 SEC.        1.225 ADJ.
12.33.16.CM       81.467 KWS.        4.972 ADJ.
12.33.16.EC        1.935 KWS.         .058 ADJ.
12.33.16.SS                          8.819
12.33.16.PP       11.098 SEC.     DATE 01/16/79
12.33.16.EJ  END OF JOB, GS
```

Figure D-2. Sequenced Input Example (Sheet 6 of 6)

```
FORTRAN 4 TO 5                    FORTRAN CONVERSION PROGRAM      F45 1.0          83/02/08. 15.56.10.      PAGE      1

                      PROGRAM SAINT27(TAPE6)                                                    ST022       2
                C     FORTRAN IV TEST 24. COMMON AND EQUIVALENCE STATEMENTS.                    ST022       3
                C     TEST24 FIV024 TESTS THE FOLLOWING FEATURES:                               ST022       4
                C     (1) TWO CONSECUTIVE SLASHES SAME AS BLANK COMMON,                         ST022       5
              5 C     (2) BLOCK NAME APPEARS MORE THAN ONCE IN COMMON STATEMENT,                ST022       6
                C     (3) ARRAY DECLARATOR IN COMMON STATEMENT,                                 ST022       7
                C     (4) ARRAY ELEMENT NAME IN EQUIVALENCE STATEMENT,                          ST022       8
                C     (5) ARRAY ELEMENT OF 3-DIMENSIONAL ARRAY APPEARS WITH ONLY ONE            ST022       9
                C         SUBSCRIPT IN EQUIVALENCE STATEMENT,                                   ST022      10
             10 C     (6) TWO QUANTITIES EQUIVALENCED AS RESULT OF JOINT THIRD QUANTITY,        ST022      11
                C     (7) TWO-STORAGE QUANTITY EQUIVALENCED TO ONE-STORAGE QUANTITY,            ST022      12
                C     (8) QUANTITY EQUIVALENCED TO A QUANTITY IN COMMON IS ASSIGNED TO          ST022      13
                C         COMMON,                                                               ST022      14
                C     (9) EQUIVALENCE STATEMENT CAUSES EXTENSION OF COMMON BLOCK.               ST022      15
             15       INTEGER P                                                                 ST022      16
                      DOUBLE PRECISION DP1, DP2, DP3                                            ST022      17
                      DIMENSION G(4)                                                            ST022      18
                      COMPLEX C1, C2                                                            ST022      19
                      REAL L1                                                                   ST022      20
             20       COMMOND, DP1 / L1 / X,Y,C1 // B / L1 / DP3                                ST022      21
                      COMMON // A(2,2,2) / L2 / U,V,W                                           ST022      22
                      EQUIVALENCE (I,J), (L1,F), (J,K), (C2,T)                                  ST022      23
                      EQUIVALENCE (DP2,DP1), (A(7),G(1))                                        ST022      24
                D     P = 6                                                                     ST022      25
             25       D = 1E+0                                                                  ST022      26
                      DP1 = 1.0D+0                                                              ST022      27
                      B = 1E+0                                                                  ST022      28
                      DO 1 KK=1,2                                                               ST022      29
                      DO 1 JJ=1,2                                                               ST022      30
             30       DO 1 II=1,2                                                               ST022      31
                1     A(II,JJ,KK) = 1.0                                                         ST022      32
                      G(3) = 1.0                                                                ST022      33
                      G(4) = 1.0                                                                ST022      34
                      I = 1                                                                     ST022      35
             35       L1 = 1.0                                                                  ST022      36
                      C2 = (1.0,1.0)                                                            ST022      37
                      WRITE (P,2)                                                               ST022      38
                2     FORMAT ( 44H1FIV024              COMMON AND EQUIVALENCE. //               ST022      39
                     196H QUANTS. EQUIVALENCED IN EACH PROGRAM UNIT ARE SET UNDER ONE NA        ST022      40
             40      2ME, REFERENCED UNDER EQUIV. NAME. /93H QUANTITIES COMMON TO MAIN          ST022      41
                     3PGM. AND TO SUBPGM. ARE SET TO ONE IN MAIN PGM.O O/P FROM SUBPGM./        ST022      42
                     462H THEN, THESE QUANTS. ARE NEGATED IN SUBPGM.O O/P FROM MAINPGM.)        ST022      43
                      WRITE (P,3) J, K, F, T, G(1), G(2), DP2                                   ST022      44
                3     FORMAT( 16HOFROM MAIN PROG.  /3H J=, I2, 3H,K=, I2, 3H,F=, F4.1,          ST022      45
             45      13H,T=, F4.1, 6H,G(1)=, F4.1, 6H,G(2)=, F4.1, 5H,DP2=, D8.1 )             ST022      46
                      CALL SUB1X (P)                                                            ST022      47
                      WRITE (P,4) D, DP1, B, A(1,1,1), X, Y, C1, DP3                            ST022      48
                4     FORMAT ( 16HOFROM MAIN PROG< / 3H D=, F4.1, 5H,DP1=, D8.1,                ST022      49
                     13H,B=, F4.1, 10H,A(1,1,1)=, F4.1, 3H,X=, F4.1/ 3H Y=, F4.1, 5H,C1=       ST022      50
             50      2( F4.1, 1H, F4.1, 6H),DP3=, D8.1 )                                       ST022      51
                      D = 1.0                                                                   ST022      52
                      DP1 = 1.0D+0                                                              ST022      53
                      CALL SUB2X (P)                                                            ST022      54
                      WRITE (P,5) D, DP1, U, V, W                                               ST022      55
             55       FORMAT( 17HOFROM MAIN PROG. / 3H D=, F4.1, 5H,DP1=, D8.1,                 ST022      56
                     1 3H,U=, F4.1, 3H,V=, F4.1, 3H,W=, F4.1 )                                  ST022      57
                      STOP                                                                      ST022      58
                      END                                                                       ST022      59
                      SUBROUTINE SUB1X(C)                                                       ST022      60
```

Figure D-3. COMPILE File Input Example (Sheet 1 of 6)

```
FORTRAN 4 TO 5                    FORTRAN CONVERSION PROGRAM      F4S 1.0          83/02/08. 15.56.10.      PAGE      2

          60                COMPLEX CX                                                          ST022      61
                            DOUBLE PRECISION DZ, DX                                             ST022      62
                            COMMON R, DZ, S, P / L1 / X1, Y1, CX, DX                            ST022      63
                            EQUIVALENCE (E1,E2)                                                 ST022      64
                            E1 = 1.0                                                            ST022      65
          65                WRITE (I,2) R, DZ, S, P X1, Y1, CX, DX, E2                          ST022      66
                     2      FORMAT( 14HOFROM SUBPGM1_  /3H R=, F4.1, 4H,DZ=, D8.1, 3H,S=,       ST022      67
                           1F4.1, 3H,P=, F4.1, 4H,X1=, F4.1, 4H,Y1=, F4.1 /                     ST022      68
                           25H C1=( F4.1, 1H, F4.1, 5H),DX=, 08.1, 4H,E2=, F4.1 )               ST022      69
                            R = -1.0                                                            ST022      70
          70                DZ = -1.0D+0                                                        ST022      71
                            S = -1.0                                                            ST022      72
                            P = -1.0                                                            ST022      73
                            X1 = -1.0                                                           ST022      74
                            Y1 = -1.0                                                           ST022      75
          75                CX = (-1.0,-1.0)                                                    ST022      76
                            DX = -1.0D+0                                                        ST022      77
                            RETURN                                                              ST022      78
                            END                                                                 ST022      79
                            SUBROUTINE SUB2X(C)                                                 ST022      80
          80                DOUBLE PRECISION DOS                                                ST022      81
                            COMMON Z, DOS / L2 / U2, V2, W2                                     ST022      82
                            EQUIVALENCE (I1,I2)                                                 ST022      83
                            I1 = 1                                                              ST022      84
                            WRITE (I,1) Z, OOS, U2, V2, W2, I2                                  ST022      85
          85          1     FORMAT ( 14HOFROM SUBPGM2_  /3H Z=, F4.1, 5H,DOS=, D8.1, 4H,U2=,    ST022      86
                           1F4.1, 4H,V2=, F4.1, 4H,W2=, F4.1, 4H,I2=, I2 )                      ST022      87
                            Z = -1.0                                                            ST022      88
                            DOS = -1.0D+0                                                       ST022      89
                            U2 = -1.0                                                           ST022      90
          90                V2 = -1.0                                                           ST022      91
                            W2 = -2.0                                                           ST022      92
                            RETURN                                                              ST022      93
                            END                                                                 ST022      94
                            BLOCK DATA                                                          ST022      95
          95                COMMON / L1 / XBD, YBD, CBD, DBD / L2 / UBD, VBD, WBD               ST022      96
                            COMPLEX CBD                                                         ST022      97
                            DOUBLE PRECISION DBD                                                ST022      98
                            DATA XBD, YBD, CBD, DBD, UBD, VBD, WBD / 2*1.0,(1.0,1.0),1.0D+0,    ST022      99
                           1 3*1.0 /                                                            ST022     100
         100                END                                                                 ST022     101

        NO. OF CARDS=  100        NO. OF ERRORS=  0
```

Figure D-3. COMPILE File Input Example (Sheet 2 of 6)

```
                    SAINT27          FORTRAN CONVERSION PROGRAM      F45 1.0      01/16/79  12.20.37.      PAGE      1
/IDENT UPDMDFY
/D,ST022.30
C*F45V1P0*                                                                                              -I-
      PROGRAM SAINT27(TAPE6)                                                                            -R-
/D,ST022.50
      REAL ZZZ112(8)                                                                                    -I-
      EQUIVALENCE (ZZZ112(1),A(1,1,1))                                                                  -I-
      EQUIVALENCE ( DP2,DP1), ( ZZZ112(7), G(1) )                                                       -R-
          ***A61*** TEMPORARY NAME SUBSTITUTED FOR OVERSUBSCRIPTED ARRAY
/D,ST022.75,ST022.77
4     FORMAT ( 16H0FROM MAIN PROG≤ / 3H D=, F4.1, 5H,DP1=, D8.1,                                        -R-
     13H,B=, F4.1, 10H,A(1,1,1)=, F4.1, 3H,X=, F4.1/ 3H Y=,F4.1,5H,C1=(                                 -R-
     2 ,F4.1, 1H, ,F4.1, 6H),DP3=, D8.1 )                                                               -R-
          ***A43*** PUNCTUATION PLACED AFTER  X, H OR DELIMTED DESCRIPTORS
```

```
MESSAGE SUMMARY   SAINT27          FORTRAN CONVERSION PROGRAM      F45 1.0      01/16/79  12.20.37.      PAGE      2

 CODE  FREQUENCY

 A43         1
 A61         1
```


Figure D-3. COMPILE File Input Example (Sheet 3 of 6)

```
                    BLKDATA          FORTRAN CONVERSION PROGRAM      F45 1.0      01/16/79  12.20.37.      PAGE      1

/D,ST022.87
C*F45V1P0*                                                                                              -I-
      COMPLEX CX                                                                                        -R-
/D,ST022.93,ST022.95
2     FORMAT( 14HOFROM SUBPGM1≤ /3H R=, F4.1, 4H,DZ=, D8.1, 3H,S=,                                      -R-
     1F4.1, 3H,P=, F4.1, 4H,X1=, F4.1, 4H,Y1=, F4.1  /                                                  -R-
     25H C1=( ,F4.1, 1H, ,F4.1, 5H),DX=, D8.1, 4H,E2=, F4.1 )                                           -R-
          ***A43*** PUNCTUATION PLACED AFTER  X, H OR DELIMTED DESCRIPTORS
```

```
MESSAGE SUMMARY   BLKDATA          FORTRAN CONVERSION PROGRAM      F45 1.0      01/16/79  12.20.37.      PAGE      2

 CODE  FREQUENCY

 A43         1
```

```
                    BLKDATA          FORTRAN CONVERSION PROGRAM      F45 1.0      01/16/79  12.20.37.      PAGE      1

/D,ST022.107
C*F45V1P0*                                                                                              -I-
      DOUBLE PRECISION DOS                                                                              -R-
```

Figure D-3. COMPILE File Input Example (Sheet 4 of 6)

```
                    BLKDATA            FORTRAN CONVERSION PROGRAM      F45 1.0        01/16/79  12.20.37.      PAGE      1
/D,ST022.121
C*F45V1P0*                                                                                          -I-
      BLOCK DATA                                                                                    -R-

  -DIAGNOSTICS-

  -STATISTICS-

    CARDS         96          CP SECS     3.098          CARDS/MIN      1859
```

```
/IDENT UPDMDFY
/D,ST022.30
C*F45V1P0*
      PROGRAM SAINT27(TAPE6)
/D,ST022.50
      REAL ZZZ112(8)
      EQUIVALENCE (ZZZ112(1),A(1,1,1))
      EQUIVALENCE ( DP2,DP1), ( ZZZ112(7), G(1) )
/D,ST022.75,ST022.77
4     FORMAT ( 16H0FROM MAIN PROG≤ / 3H D=, F4.1, 5H,DP1=, D8.1,
     13H,B=, F4.1, 10H,A(1,1,1)=, F4.1, 3H,X=, F4.1/ 3H Y=,F4.1,5H,C1=(
     2 ,F4.1, 1H, ,F4.1, 6H),DP3=, D8.1 )
/D,ST022.87
C*F45V1P0*
      COMPLEX CX
/D,ST022.93,ST022.95
2     FORMAT( 14H0FROM SUBPGM1≤ /3H R=, F4.1, 4H,DZ=, D8.1, 3H,S=,
     1F4.1, 3H,P=, F4.1, 4H,X1=, F4.1, 4H,Y1=, F4.1  /
     25H C1=( ,F4.1, 1H, ,F4.1, 5H),DX=, D8.1, 4H,E2=, F4.1 )
/D,ST022.107
C*F45V1P0*
      DOUBLE PRECISION DOS
/D,ST022.121
C*F45V1P0*
      BLOCK DATA
```

Figure D-3. COMPILE File Input Example (Sheet 5 of 6)

```
  MFS  NB1-  CYB74-SN108    5C/R0B    11/14/78
12.16.57.EHFCPHG  FROM    /GS
12.16.57.IP  00000128 WORDS - FILE INPUT  , DC 04
12.16.57.EHFCP (T300, P1, EC1)     018G, 7182, 4
12.16.57.3LM1040, FTNERH, X7333, SVL166.
12.17.03.ATTACH,FTNLIB,ID=CPO.
12.17.03.PFN IS
12.17.03.FTNLIB
12.17.08.PF CYCLE NO. = 007
12.17.08.LIBRARY(FTNLIB)
12.17.08.REWIND,OUTPUT.
12.17.18.ATTACH(OLDPL,FC4PL,ID=PSAUDIT)
12.17.20.  OLDPL    CY = 53
12.18.07.ATTACH(DUP1,ID=FTNERH)
12.18.10.  DUP1     CY = 2
12.18.10.UPDATE,*=/,Q,C=TAPE1,I=DUP1.
12.18.44. UPDATE COMPLETE
12.18.45.
12.18.45.    STRIP OFF JOB CONTROL CARDS.
12.18.45.
12.18.47.REWIND(TAPE1,TAPE6)
12.18.48.COPYBR(TAPE1,NULL)
12.18.50.COPY(TAPE1,TAPE6)
12.18.53.REWIND,TAPE6.
12.20.34.F45,I=TAPE6,LO=F,P=POUT,PO=M,MC=$/$,CI=U
12.20.34.PDMDFY.
12.21.41.     STOP  F45
12.21.41.REWIND,POUT.
12.21.42.COPYSBF(POUT,OUTPUT)
12.21.43. END OF INFORMATION ENCOUNTERED.
12.21.43.EXIT(S)
12.21.43.OP  00004992 WORDS - FILE OUTPUT , DC 40
12.21.43.MS     7168  WORDS  (   35840 MAX USED)
12.21.43.CPA       4.934 SEC.        4.934 ADJ.
12.21.43.CPB        .708 SEC.         .708 ADJ.
12.21.43.IO        2.289 SEC.        2.289 ADJ.
12.21.43.CM      156.326 KWS.        9.541 ADJ.
12.21.43.EC        4.053 KWS.         .123 ADJ.
12.21.43.SS                         17.597
12.21.44.PP       19.851 SEC.     DATE 01/16/79
12.21.44.EJ  END OF JOB, GS
```

Figure D-3. COMPILE File Input Example (Sheet 6 of 6)

INDEX

COMMENT SHEET

MANUAL TITLE: FORTRAN Extended Version 4 to FORTRAN Version 5
Conversion Aid Program Version 1 Reference Manual

PUBLICATION NO.: 60483000 REVISION: C

NAME:

COMPANY:

STREET ADDRESS:

CITY: STATE: ZIP CODE:

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

__ Please reply __ No reply necessary

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND TAPE

TAPE

TAPE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division

215 Moffett Park Drive
Sunnyvale, California 94086

FOLD

FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S

CONTROL DATA CORPORATION